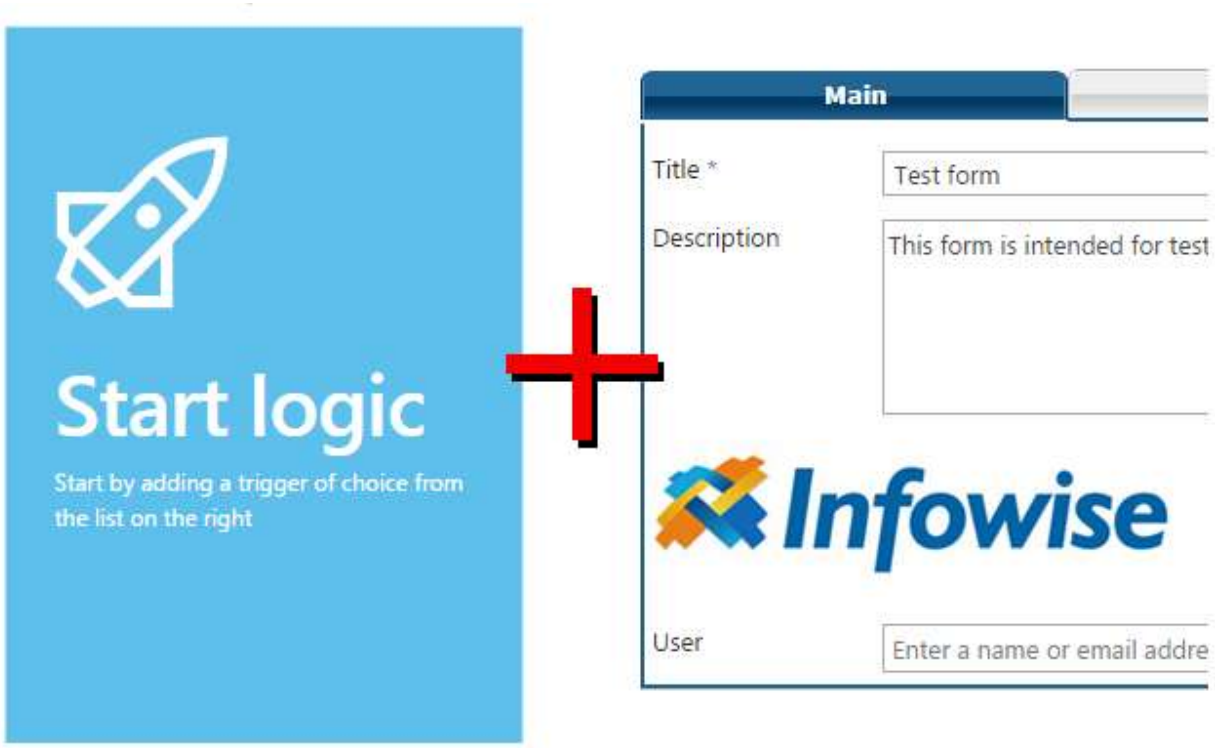

Start logic
Start by adding a trigger of choice from the list on the right
Main
Infowise

Title * Test form

Description This form is intended for test

User Enter a name or email addre

Easily Harness the power of Azure in your SharePoint Forms by integrating Infowise Ultimate Forms and Azure Logic Apps

Sales: sales@infowisesolutions.com
Support Issues: support@infowisesolutions.com
General Inquiry: info@infowisesolutions.com

1920 Yonge Street, Suite 200
Toronto, ON, M4S 3E2, Canada
Phone: +1.416.572.7755
Fax: +1.416.572.7501

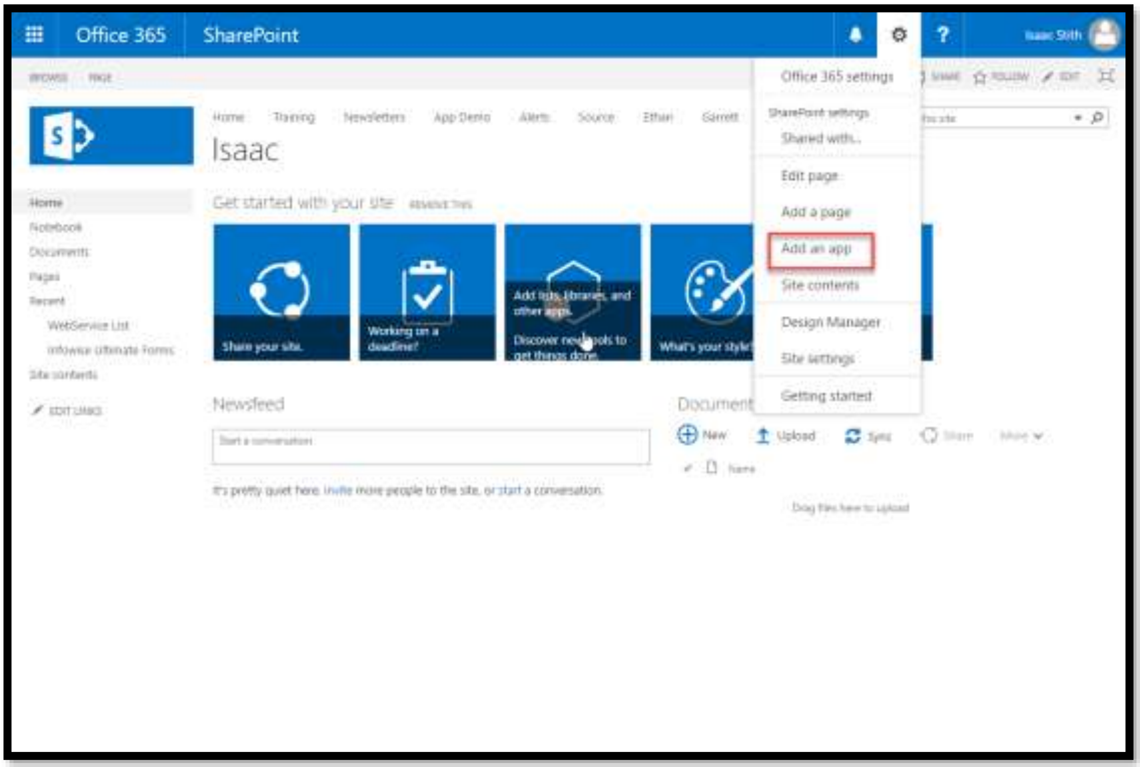
Integrating Azure Logic Apps with Infowise Ultimate Forms

One of the major benefits of using the Infowise forms is just how powerful and easy the forms process is with Ultimate Forms in SharePoint Online. A great example of this power and flexibility is how it is possible extending your forms in SharePoint with actions from Azure. One of the many available Azure services for integration, is the Logic Apps platform. Using preset connectors and logic, you can configure tie ins to many internet services like Twitter, LinkedIn, Dynamics CRM and more. The best part is that you can launch these logic applications from within your Infowise forms, making a seamless business process that integrates across cloud environments. What follows is a proof of concept showing the integration between SharePoint Online with Azure using Infowise Ultimate Forms.

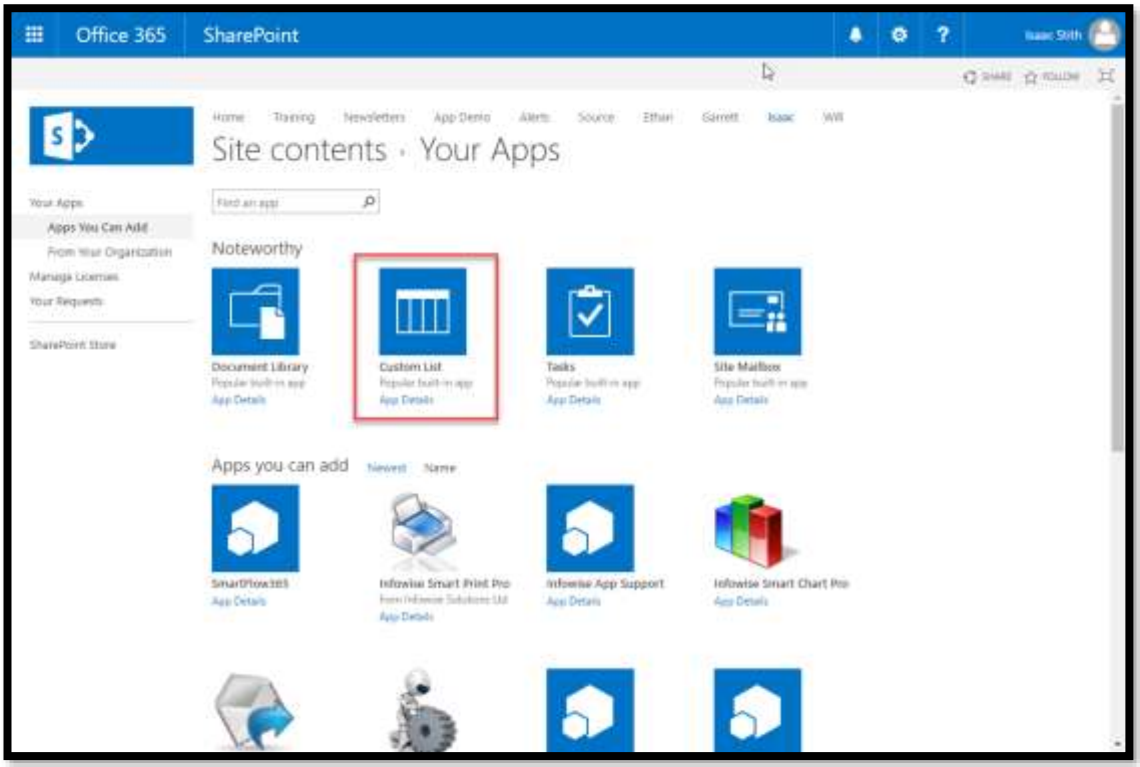
The Scenario: A team of users collaborating using a single Twitter account sharing information and interacting with users. One difficulty the team experiences comes from authenticating and managing access to the Twitter account, since each tweet requires approval before posting to company's twitter feed. This scenario consists of setting up the form in SharePoint Online in Infowise, then configuring the Action in the Azure Logic App, and finally tying it all together to publish the tweets on twitter.

Configuring SharePoint

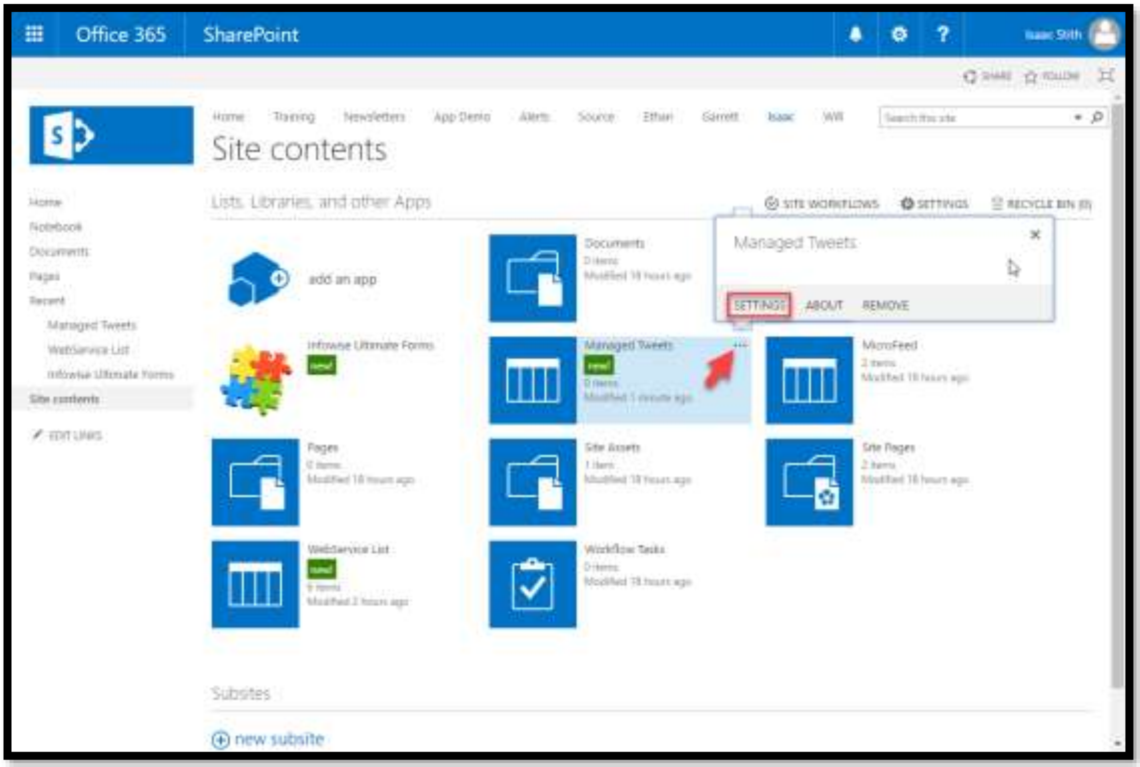
Once you are in your SharePoint environment, choose Add an App from the settings menu.



Choose the custom list option when prompted to create a new application.



Give the list a name like Managed Tweets and create the custom list.

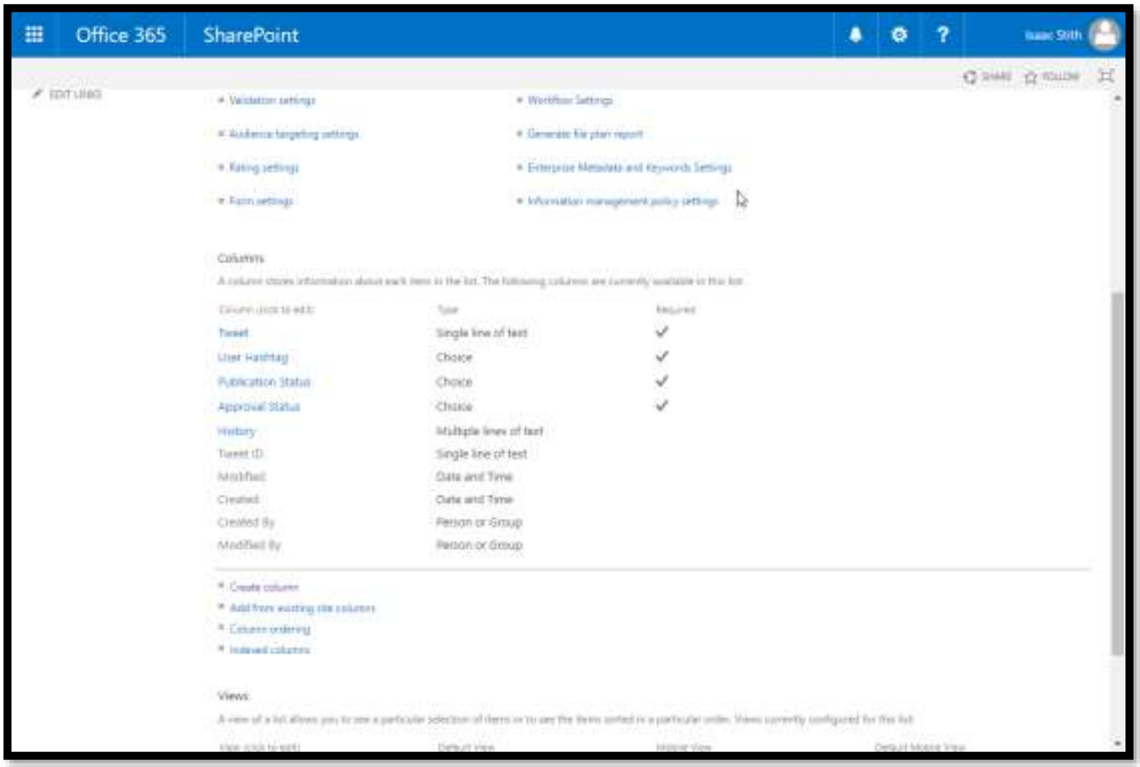


Then choose the settings button when clicking the **…** menu option as you hover over the newly created list.

The create the columns as outlined in the table below

Name	Type	Value	Required
Tweet	Rename Title Field		Yes
User Hashtag	Choice	^CS ^ACC ^MKT ^MGT ^CEO	Yes
Publication Status	Choice	Unpublished Published Error	Yes
Approval Status	Choice	Approved Rejected Pending	Yes
History	Multiple Lines of Text (plain text)		No
Tweet ID	Single Line of Text		No

The result of which, you can see in the following screenshot


Office 365
SharePoint
Isaac Stith
SHARE
FOLLOW
EDIT LINKS
Validation settings
Audience targeting settings
Rating settings
Form settings
Workflow Settings
Generate file plan report
Enterprise Metadata and Keywords Settings
Information management policy settings

Columns

A column stores information about each item in the list. The following columns are currently available in this list:

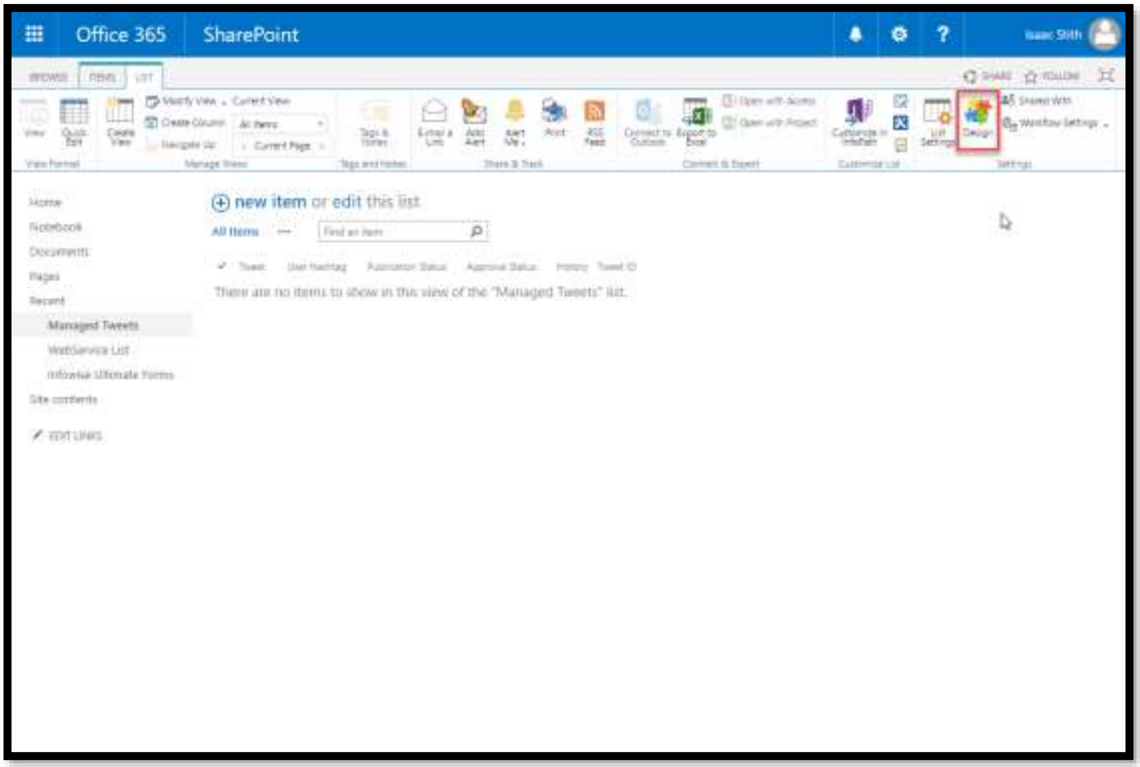
Column (click to edit)	Type	Required
Tweet	Single line of text	✓
User Hashtag	Choice	✓
Publication Status	Choice	✓
Approval Status	Choice	✓
History	Multiple lines of text	
Tweet ID	Single line of text	
Modified	Date and Time	
Created	Date and Time	
Created By	Person or Group	
Modified By	Person or Group	

Views

A view of a list allows you to see a particular selection of items or to see the items sorted in a particular order. Views currently configured for this list:

View (click to edit)	Default View	Mobile View	Default Mobile View

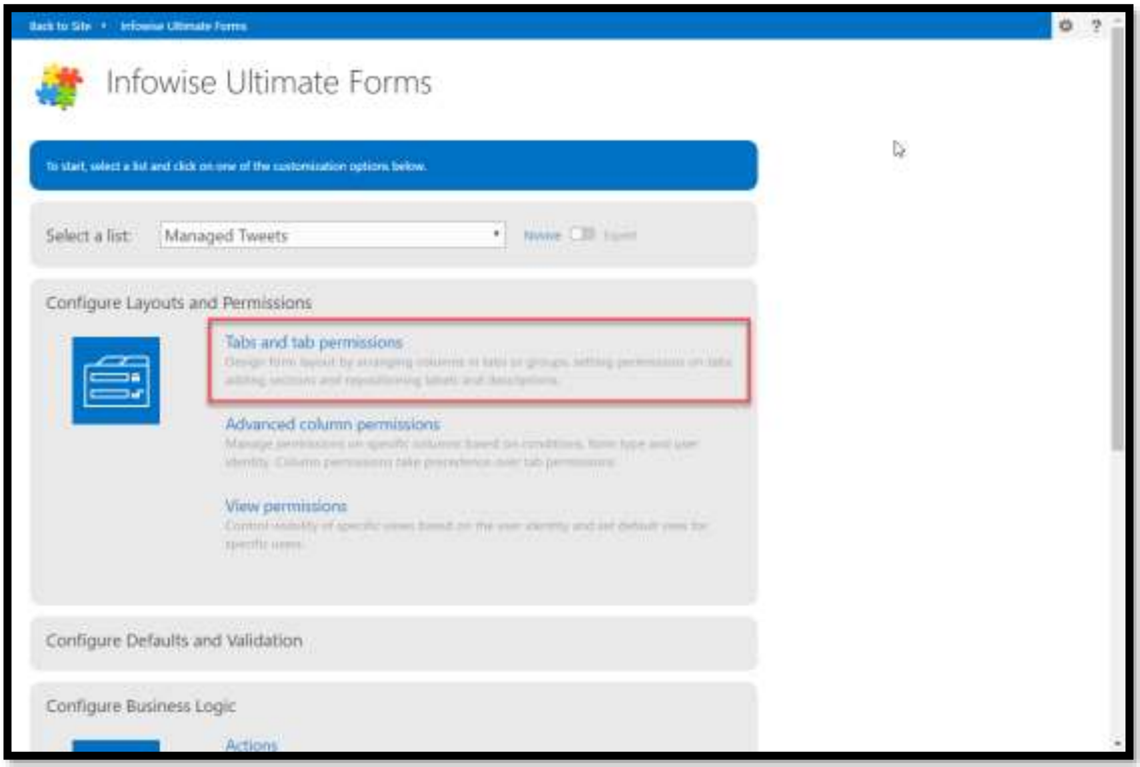
Once we have finished creating all the columns, return to the home page of the list and choose design to Launch the Infowise Ultimate Forms service.



⊕ new item or edit this list

There are no items to show in this view of the "Managed Tweets" list.

The first thing we are going to do is configure the tabs and the layouts of the form, so click the Tabs and Permissions Option



Infowise Ultimate Forms

To start, select a list and click on one of the customization options below.

Select a list: Managed Tweets

Configure Layouts and Permissions

Tabs and tab permissions
Design form layout by arranging columns in tabs or groups, setting permissions on tabs, adding sections and repositioning labels and descriptions.

Advanced column permissions
Manage permissions on specific columns based on conditions, form type and user identity. Column permissions take precedence over tab permissions.

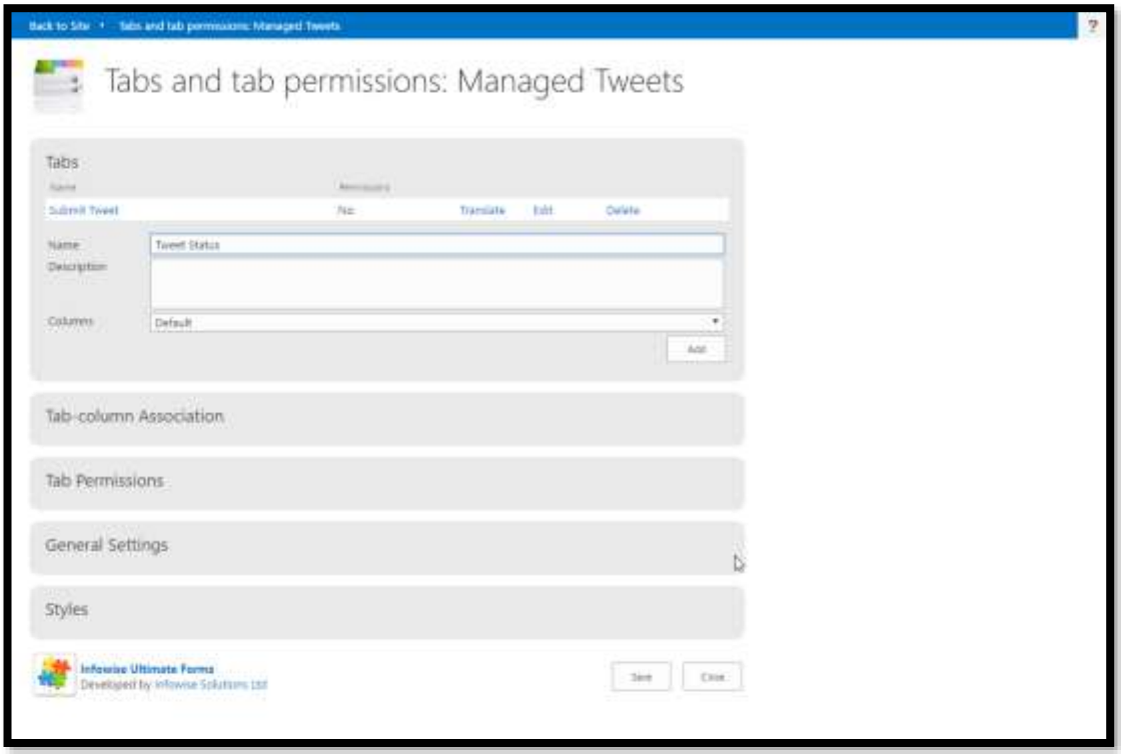
View permissions
Control visibility of specific views based on the user identity and set default view for specific users.

Configure Defaults and Validation

Configure Business Logic

Actions

Create two tabs called “Submit Tweet and Tweet Status” making sure you click Add Each Time



Tabs and tab permissions: Managed Tweets

Tabs

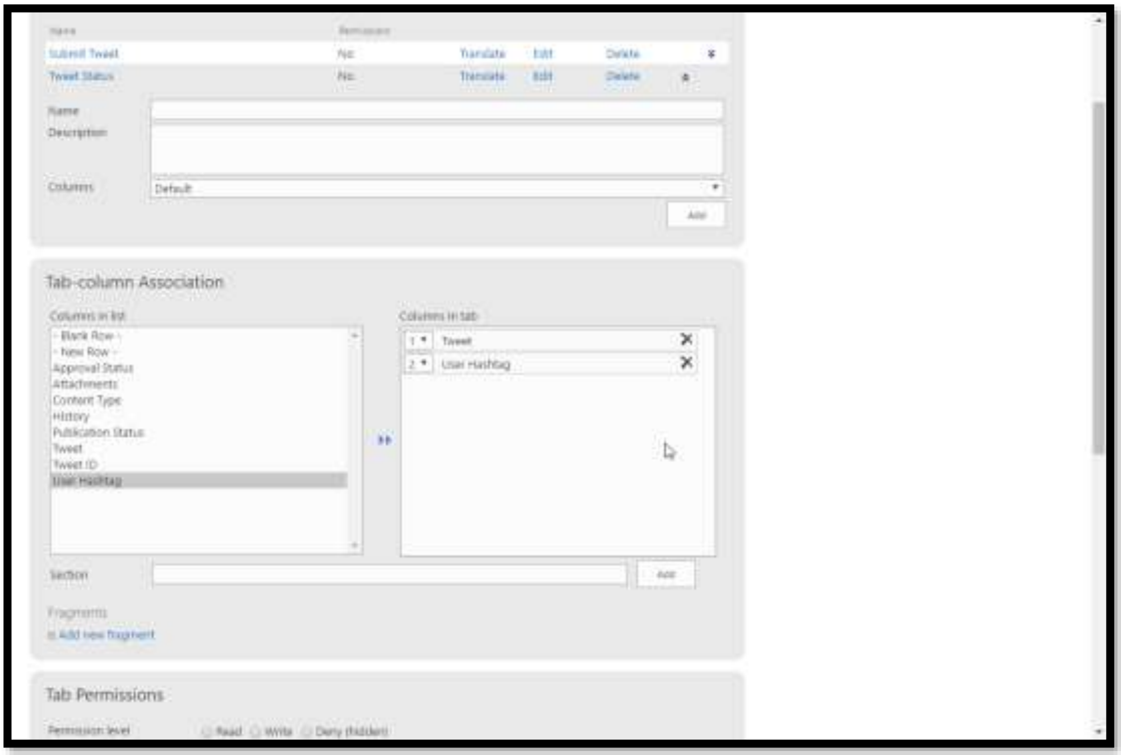
Name	Permissions			
Submit Tweet	No	Translate	Edit	Delete

Name: Tweet Status

Description:

Columns: Default

Click the Submit Tweet Tab and then expand the Tab-Column Association Box. Afterwards, add the two fields shown below Tweet, and User Hashtag to the tab



Name	Permissions			
Submit Tweet	No	Translate	Edit	Delete
Tweet Status	No	Translate	Edit	Delete

Name:

Description:

Columns: Default

Tab-column Association

Columns in list:
- Blank Row -
- New Row -
Approval Status
Attachments
Content Type
History
Publication Status
Tweet
Tweet ID
User Hashtag

Columns in tab:
1 Tweet
2 User Hashtag

Section:

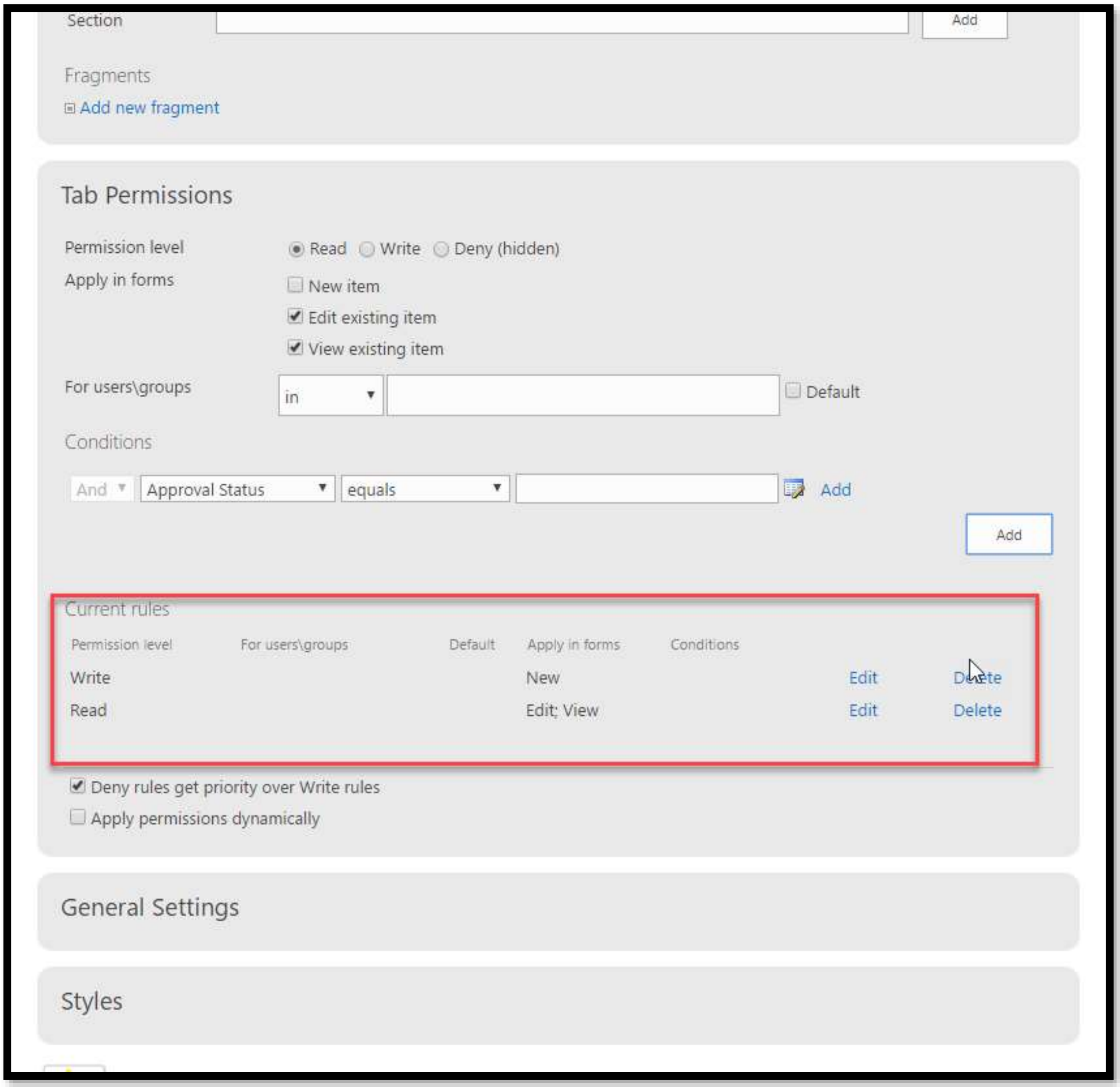
Tab Permissions

Permission level ○ Read ○ Write ○ Deny (hidden)

Then Expand the Tab Permissions Box and then add the following rules

- Write on the New Form Only
- Read on the View and Edit Forms

Since we do not want users to be able to edit the tweets after they submit them, we are going to use these permissions to restrict the forms.



Section Add

Fragments

⊞ Add new fragment

Tab Permissions

Permission level ● Read ○ Write ○ Deny (hidden)

Apply in forms
□ New item
☑ Edit existing item
☑ View existing item

For users\groups in ▾ □ Default

Conditions

And ▾ Approval Status ▾ equals ▾ Add

Add

Current rules

Permission level	For users\groups	Default	Apply in forms	Conditions		
Write			New		Edit	Delete
Read			Edit; View		Edit	Delete

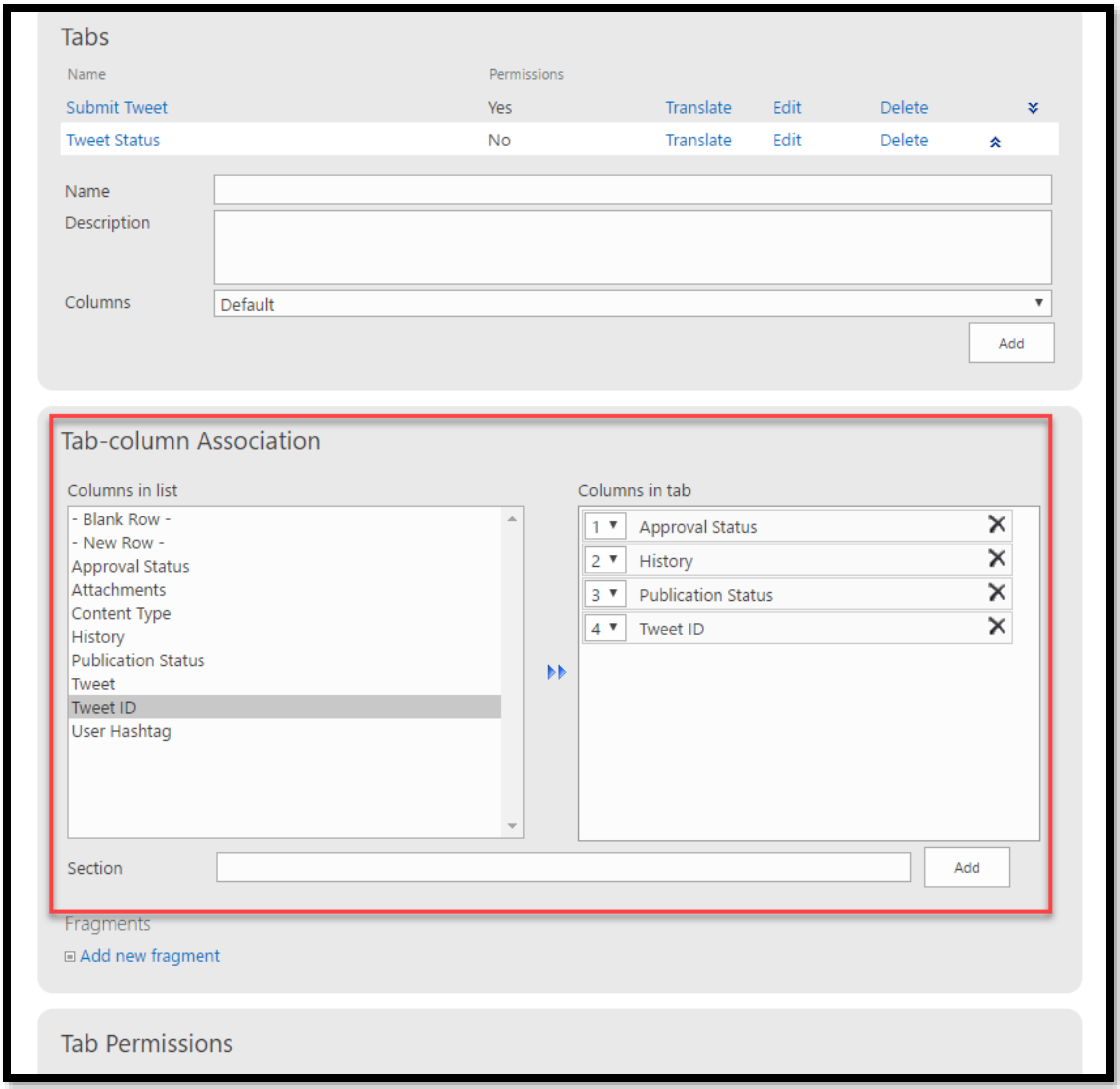
☑ Deny rules get priority over Write rules
□ Apply permissions dynamically

General Settings

Styles

We'll repeat these steps for the Tweet Status tab

In the tabs listed at the top, choose the Tweet Status tab and configure the column associations as pictured.



Tabs

Name	Permissions				
Submit Tweet	Yes	Translate	Edit	Delete	⌄
Tweet Status	No	Translate	Edit	Delete	⌃

Name

Description

Columns: Default

Tab-column Association

Columns in list:
- Blank Row -
- New Row -
Approval Status
Attachments
Content Type
History
Publication Status
Tweet
Tweet ID
User Hashtag

Columns in tab:
1 Approval Status
2 History
3 Publication Status
4 Tweet ID

Section

Fragments

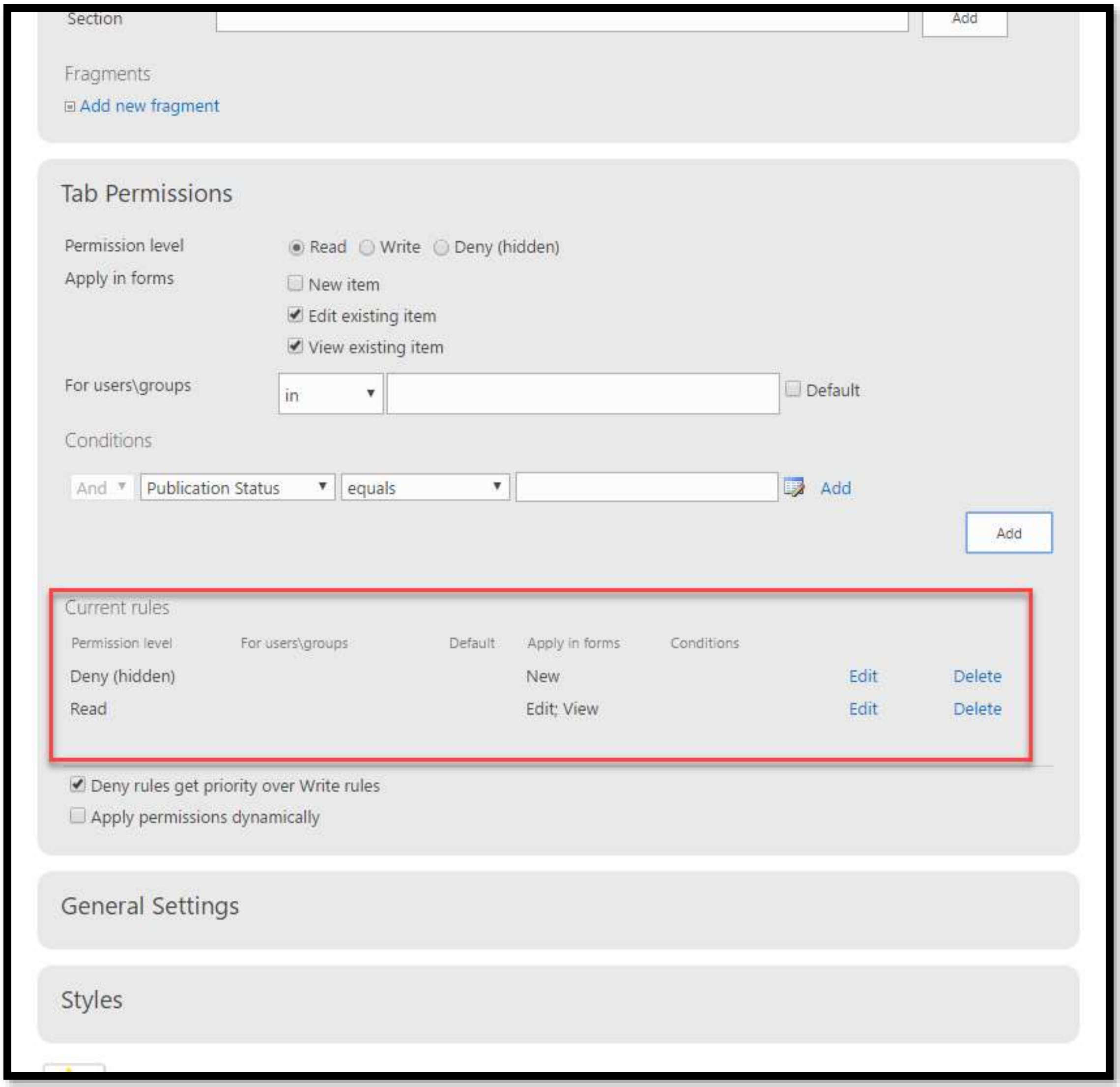
Tab Permissions

Add the following columns:

- Approval status
- History
- Publication status
- Tweet ID

Then Expand the Tab Permissions box to set the permissions.

Since we want to hide this tab for new submissions, we will deny permission to the tab and since we only want to view the content in the other fields, we will make them read only.



Section

Fragments

Tab Permissions

Permission level ● Read ○ Write ○ Deny (hidden)

Apply in forms
☐ New item
☑ Edit existing item
☑ View existing item

For users\groups in ▾ ☐ Default

Conditions

And ▾ Publication Status ▾ equals ▾

Current rules

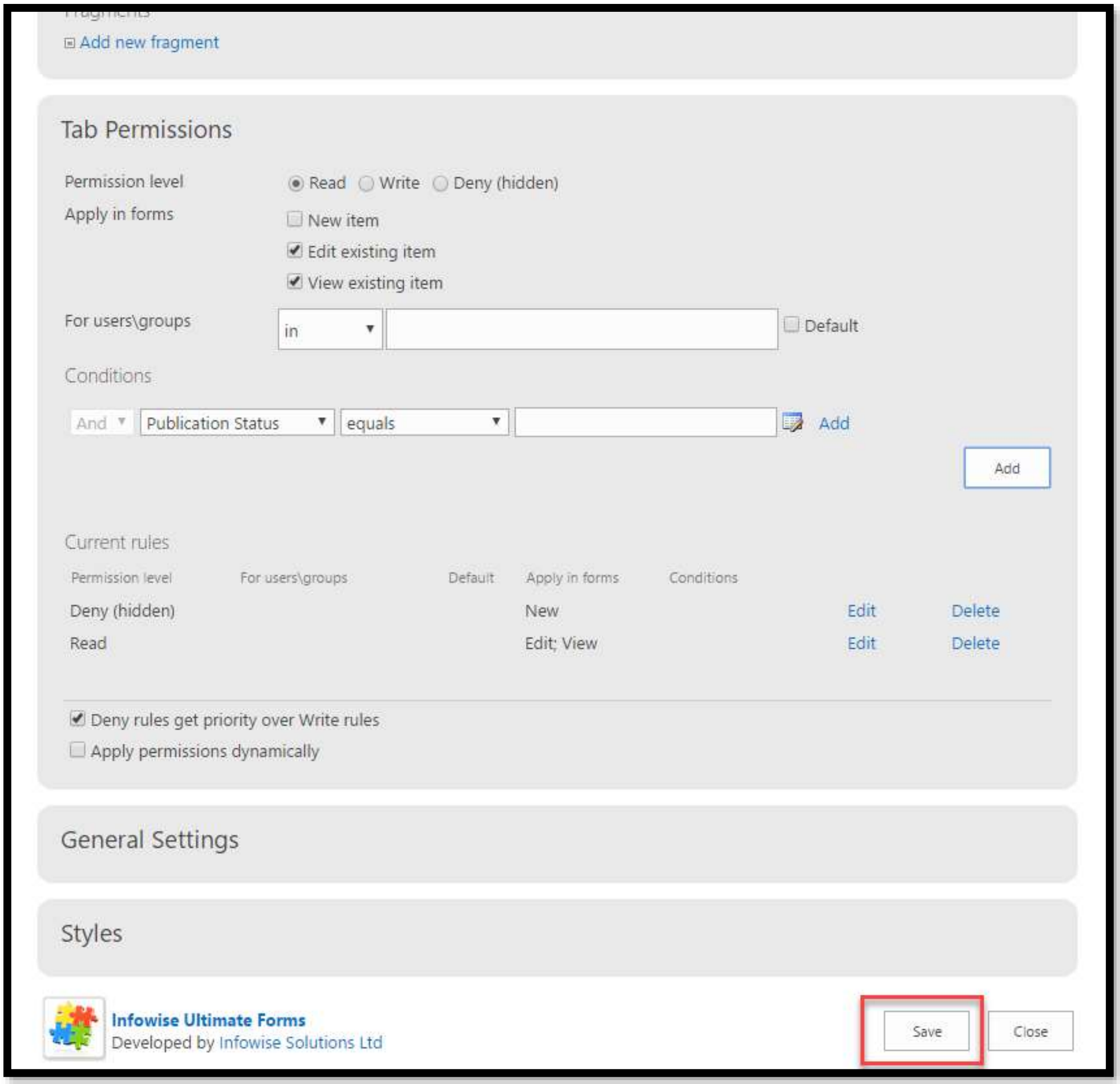
Permission level	For users\groups	Default	Apply in forms	Conditions		
Deny (hidden)			New		Edit	Delete
Read			Edit; View		Edit	Delete

☑ Deny rules get priority over Write rules
☐ Apply permissions dynamically

General Settings

Styles

After you are done with the permissions, make sure you click save



Fragments

⊞ Add new fragment

Tab Permissions

Permission level ● Read ○ Write ○ Deny (hidden)

Apply in forms

□ New item

☑ Edit existing item

☑ View existing item

For users\groups in ▾ □ Default

Conditions

And ▾ Publication Status ▾ equals ▾ Add

Add

Current rules

Permission level	For users\groups	Default	Apply in forms	Conditions		
Deny (hidden)			New		Edit	Delete
Read			Edit; View		Edit	Delete

☑ Deny rules get priority over Write rules

□ Apply permissions dynamically

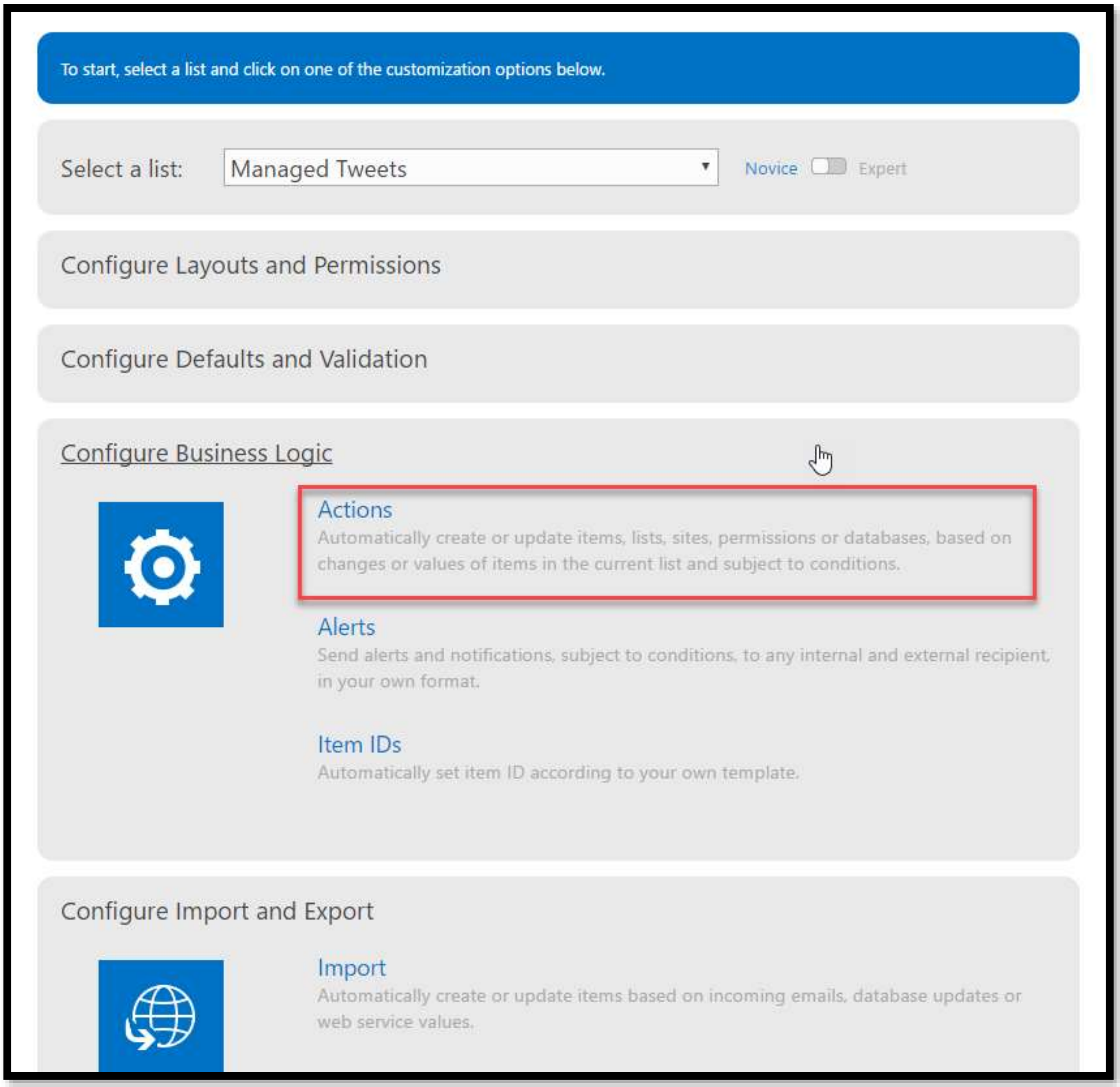
General Settings

Styles

Infowise Ultimate Forms
Developed by Infowise Solutions Ltd

Save Close

Now that we have the views configured, lets continue on to creating the approval and rejection actions. Go back to the main menu of Ultimate Forms and choose actions from the Configure Business Logic box.



To start, select a list and click on one of the customization options below.

Select a list: Managed Tweets

Configure Layouts and Permissions

Configure Defaults and Validation

Configure Business Logic

Actions
Automatically create or update items, lists, sites, permissions or databases, based on changes or values of items in the current list and subject to conditions.

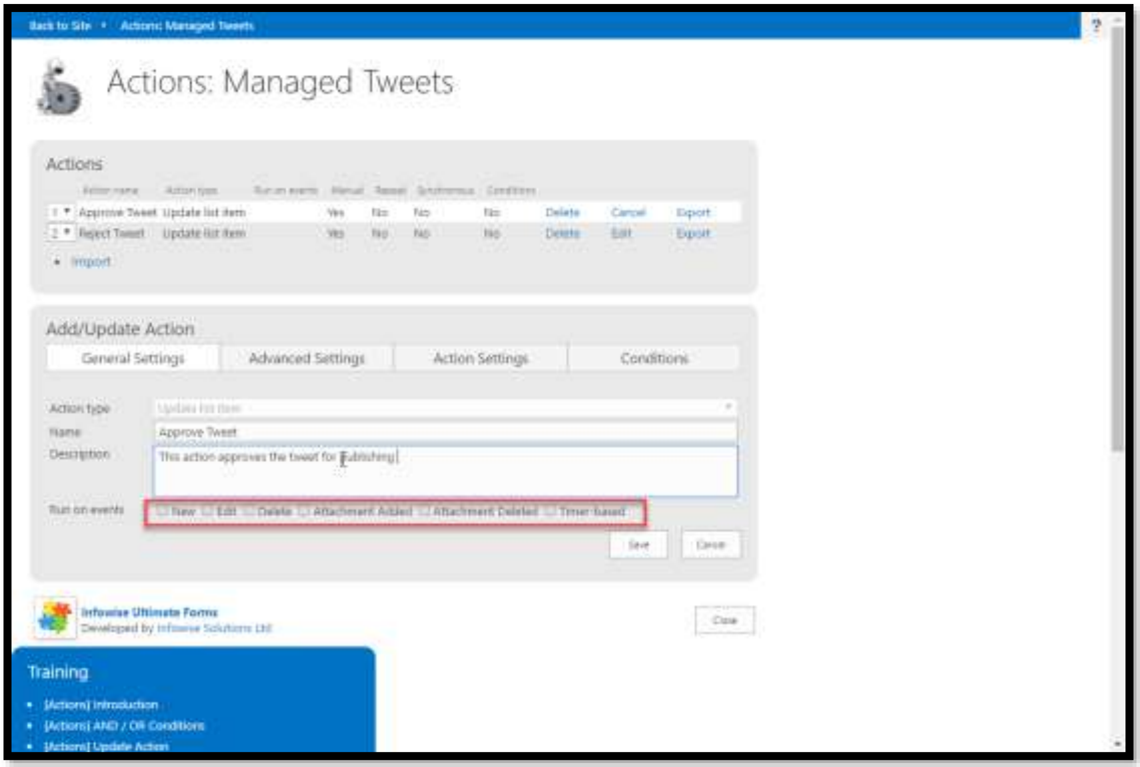
Alerts
Send alerts and notifications, subject to conditions, to any internal and external recipient, in your own format.

Item IDs
Automatically set item ID according to your own template.

Configure Import and Export

Import
Automatically create or update items based on incoming emails, database updates or web service values.

Create a new action titled Approve Tweet and give it a description



Back to Site • Actions: Managed Tweets

Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Cancel	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

Action type: Update list item

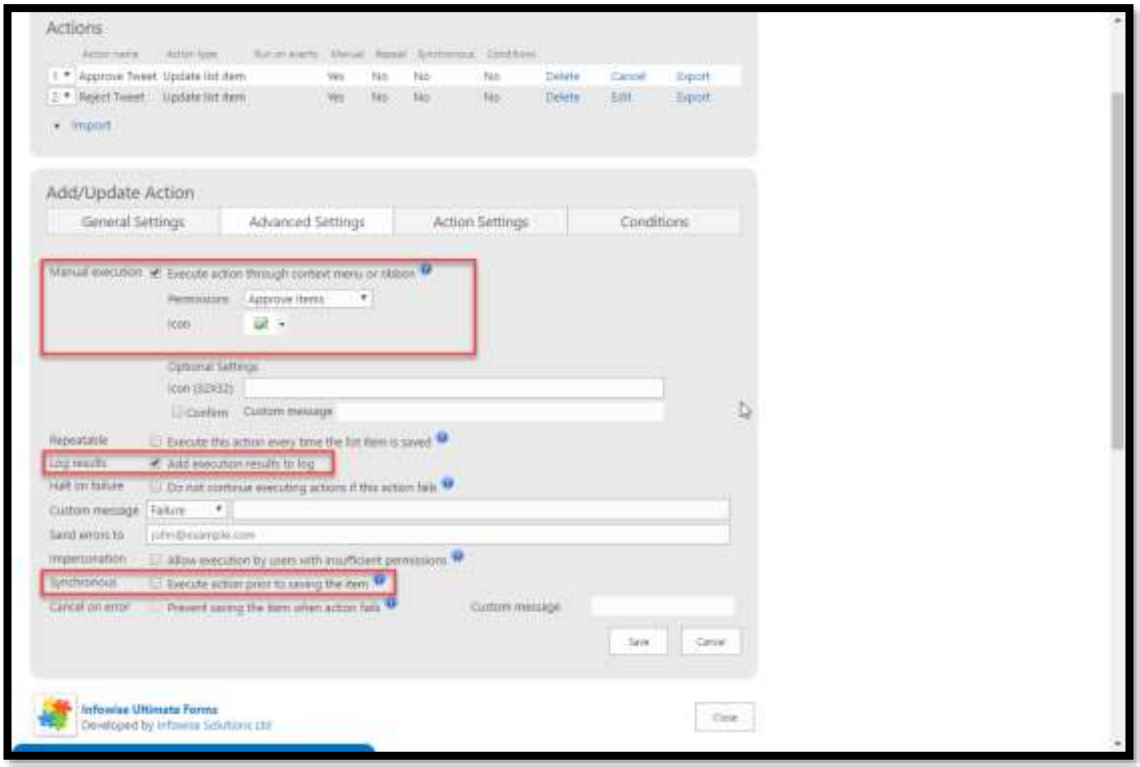
Name: Approve Tweet

Description: This action approves the tweet for publishing

Run on events: □ New □ Edit □ Delete □ Attachment Added □ Attachment Deleted □ Timer-based

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action

Make sure that there aren't any "Run on Events" checked before you continue to Advanced Settings



Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Cancel	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

Manual execution ☑ Execute action through context menu or ribbon

Permissions: Approve items

Icon

Optional Settings

Icon (32X32)

□ Confirm Custom message

Repeatable □ Execute this action every time the list item is saved

Log results ☑ Add execution results to log

Halt on failure □ Do not continue executing actions if this action fails

Custom message: Failure

Send errors to: john@example.com

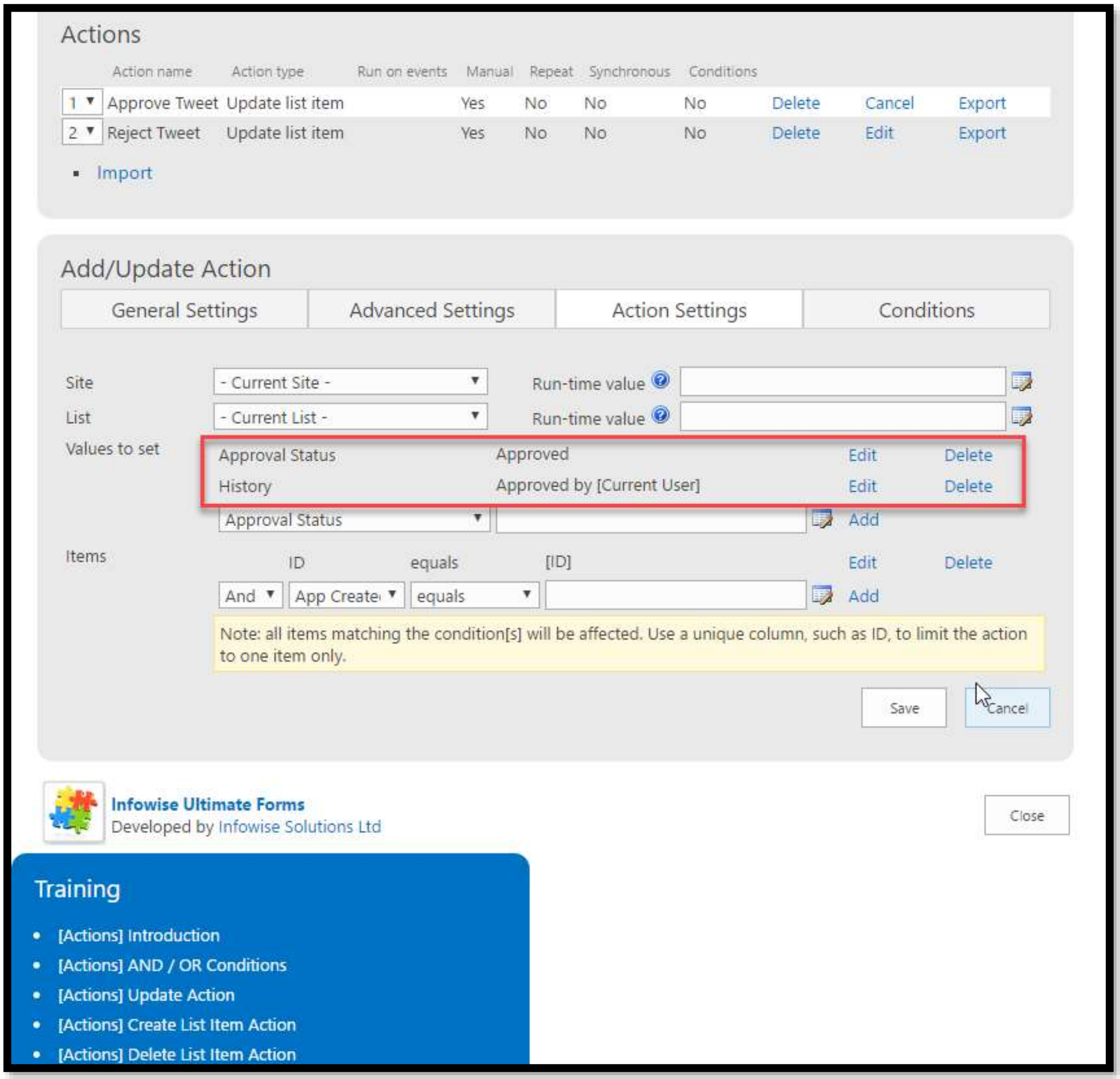
Impersonation □ Allow execution by users with insufficient permissions

Synchronous □ Execute action prior to saving the item

Cancel on error □ Prevent saving the item when action fails   Custom message

From there make sure that you choose the Manual Execution and choose the Approve Permissions for the list.

In addition, make sure that you log the results and that no synchronous events are checked.



Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1	Approve Tweet	Update list item		Yes	No	No	No	Delete	Cancel	Export
2	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

Site: - Current Site - Run-time value

List: - Current List - Run-time value

Values to set:

Approval Status	Approved	Edit	Delete
History	Approved by [Current User]	Edit	Delete

Items:

ID	equals	[ID]	Edit	Delete

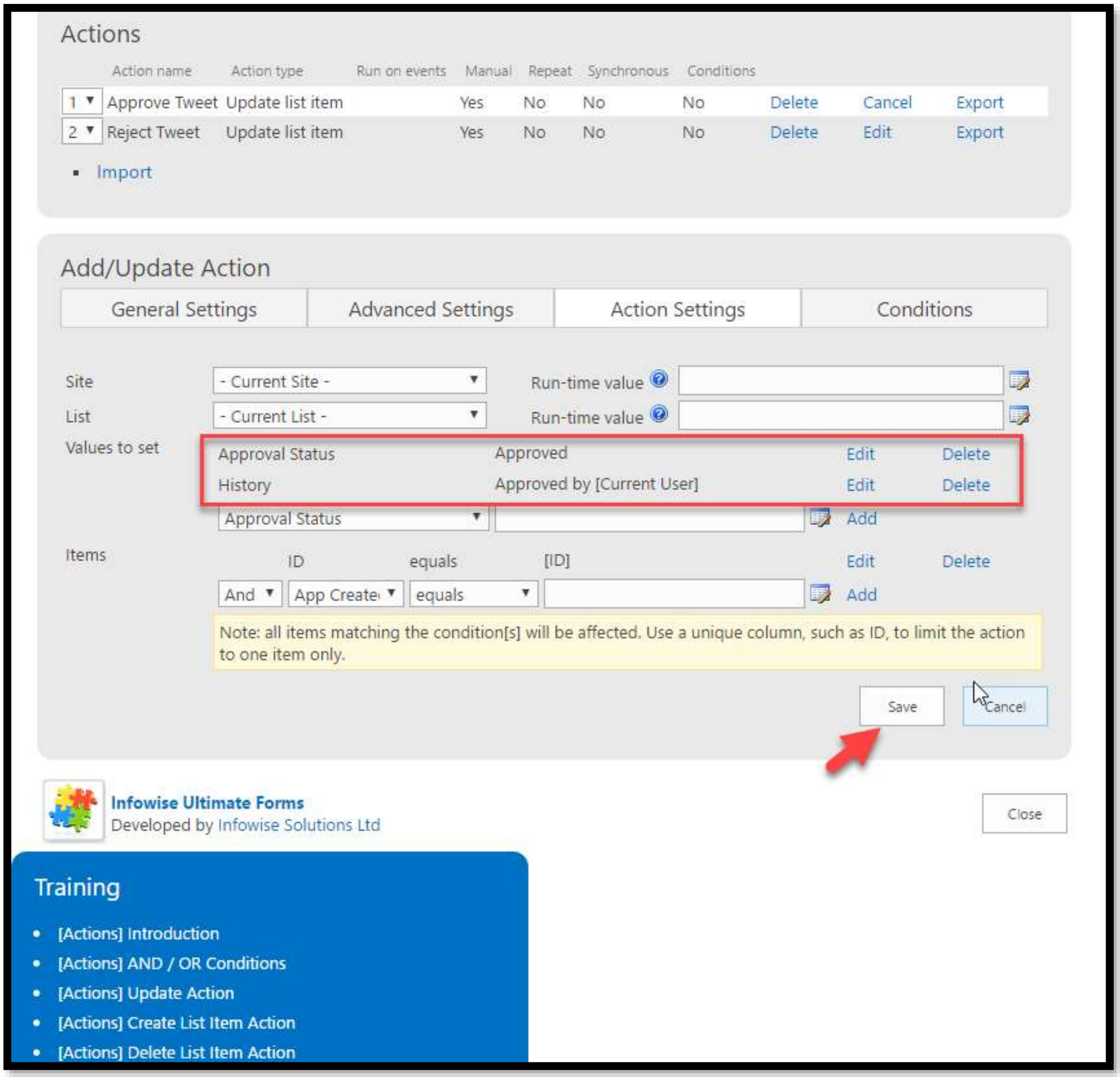
Note: all items matching the condition[s] will be affected. Use a unique column, such as ID, to limit the action to one item only.

Training

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action
- [Actions] Create List Item Action
- [Actions] Delete List Item Action

Then go to the action settings and update the following two Values

- Approval Status to Approved
- History to Approved by [Current User]



Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Cancel	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

Site: - Current Site - ▾ Run-time value

List: - Current List - ▾ Run-time value

Values to set:

Approval Status	Approved	Edit	Delete
History	Approved by [Current User]	Edit	Delete

Approval Status ▾ Add

Items:

ID	equals	[ID]	Edit	Delete

And ▾ App Create ▾ equals ▾ Add

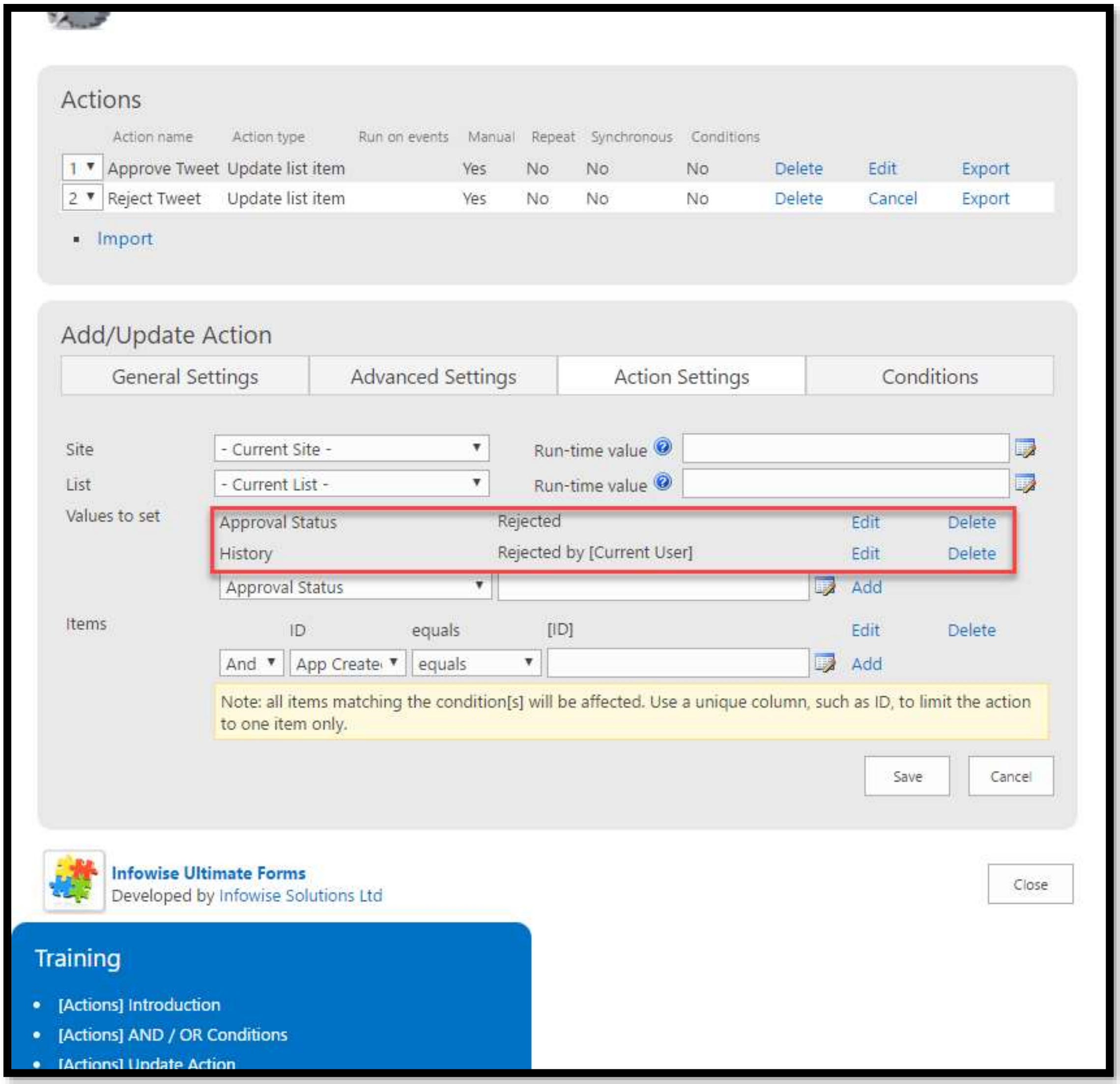
Note: all items matching the condition[s] will be affected. Use a unique column, such as ID, to limit the action to one item only.

Training

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action
- [Actions] Create List Item Action
- [Actions] Delete List Item Action

Then you can click save since we are not creating any conditions at this point.

Do the same for the Reject action, except changing the status and history information as appropriate for the action settings



Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▼	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2 ▼	Reject Tweet	Update list item		Yes	No	No	No	Delete	Cancel	Export

- Import

Add/Update Action

Site: - Current Site - ▼ Run-time value

List: - Current List - ▼ Run-time value

Values to set			
Approval Status	Rejected	Edit	Delete
History	Rejected by [Current User]	Edit	Delete

Approval Status ▼ Add

Items				
ID	equals	[ID]	Edit	Delete

And ▼ App Create ▼ equals ▼ Add

Note: all items matching the condition[s] will be affected. Use a unique column, such as ID, to limit the action to one item only.

Infowise Ultimate Forms
Developed by Infowise Solutions Ltd

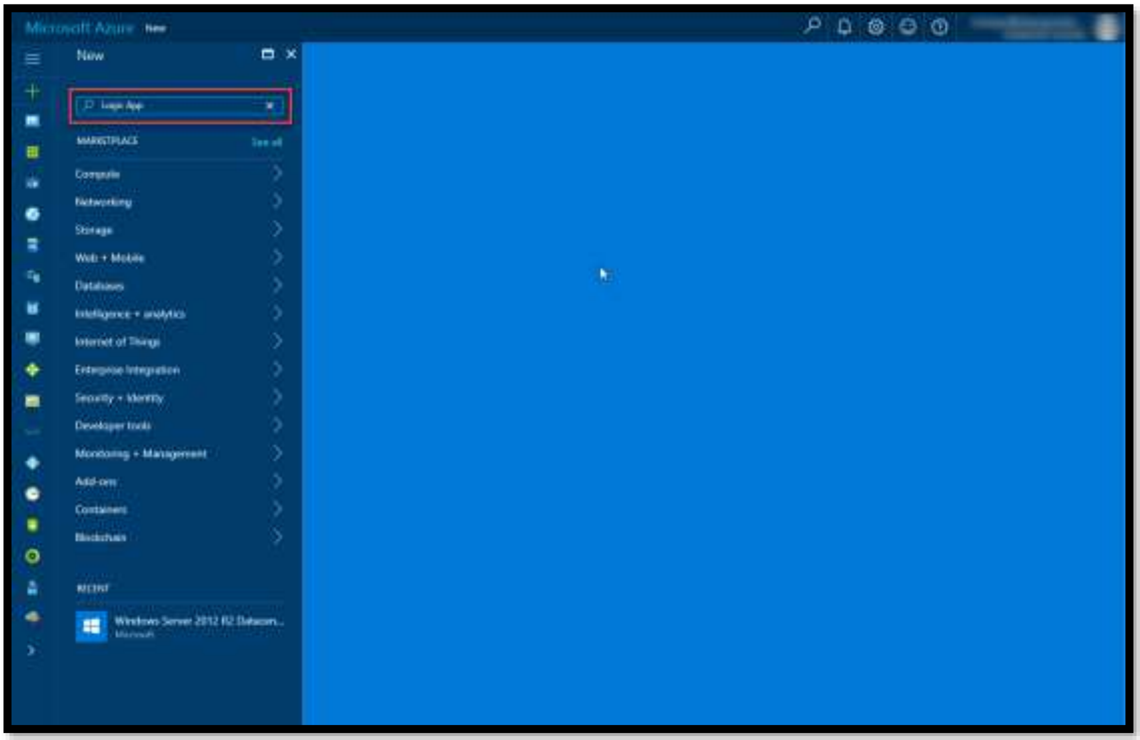
Training

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action

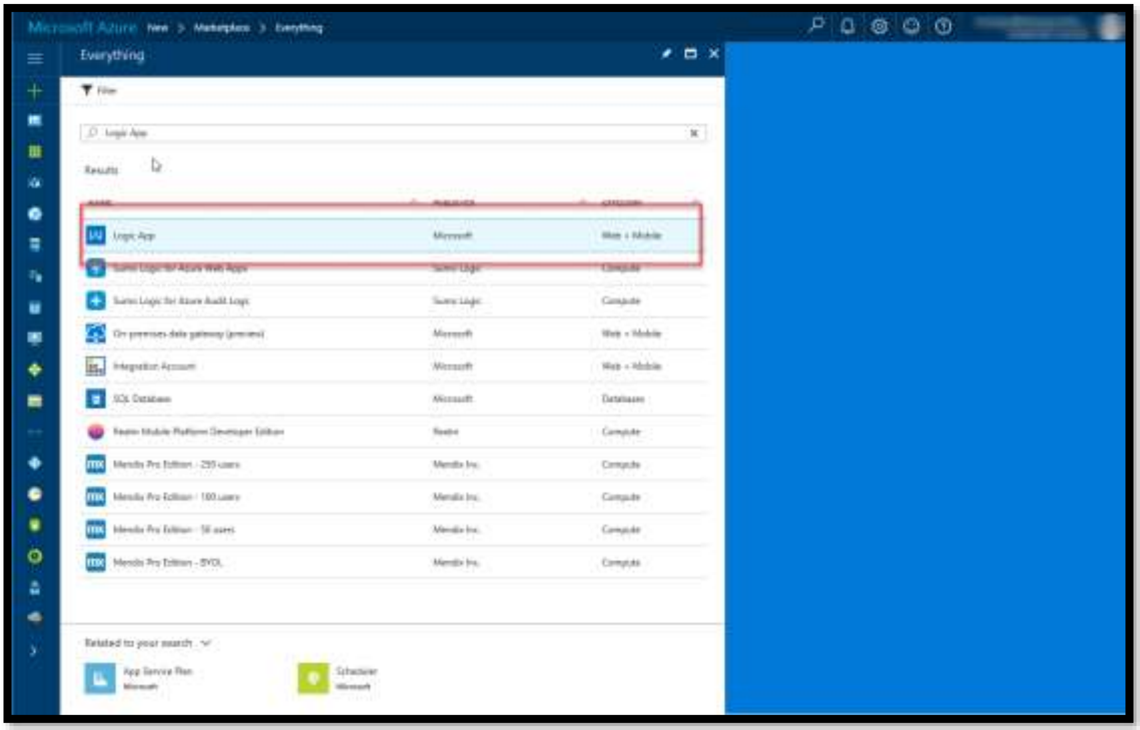
Once you have completed this, the list setup is complete and we can proceed to creating the Logic App in Azure.

Create the Logic App

In Azure Portal look for the Logic App Service and then select it

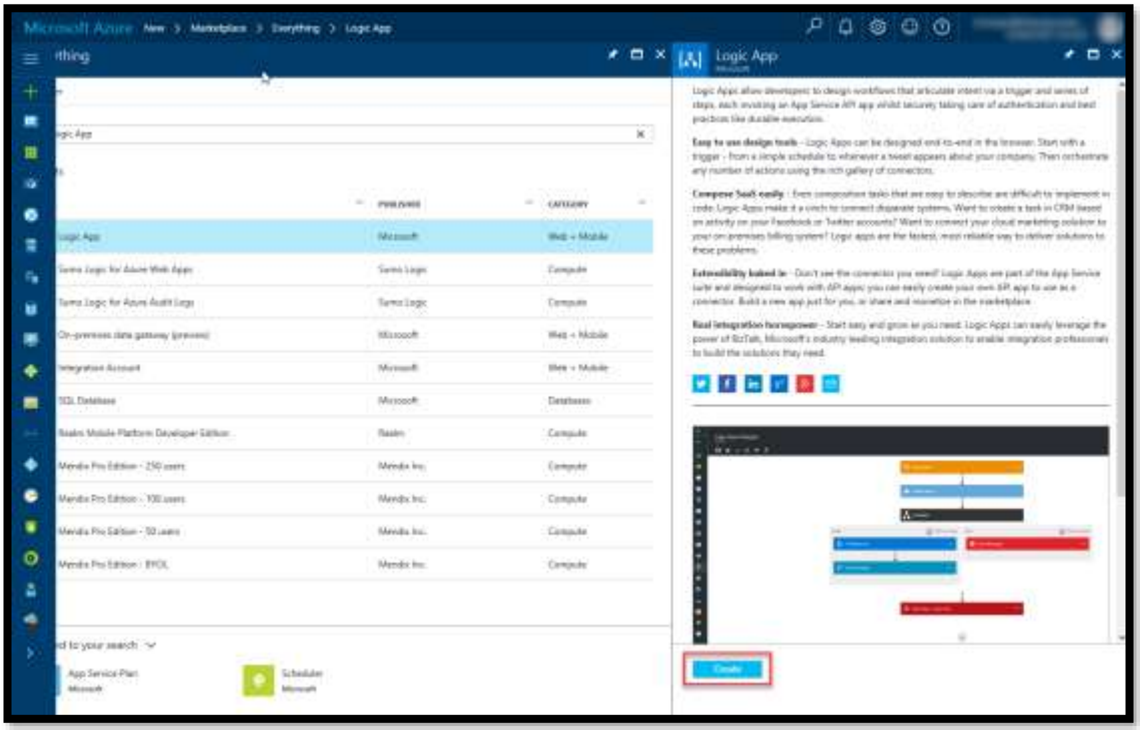


Select Logic App



NAME	PUBLISHER	CATEGORY
Logic App	Microsoft	Web + Mobile
Sumo Logic for Azure Web Apps	Sumo Logic	Compute
Sumo Logic for Azure Audit Logs	Sumo Logic	Compute
On-premises data gateway (preview)	Microsoft	Web + Mobile
Integration Account	Microsoft	Web + Mobile
SQL Database	Microsoft	Databases
Realm Mobile Platform Developer Edition	Realm	Compute
Mendix Pro Edition - 250 users	Mendix Inc.	Compute
Mendix Pro Edition - 100 users	Mendix Inc.	Compute
Mendix Pro Edition - 50 users	Mendix Inc.	Compute
Mendix Pro Edition - BYOL	Mendix Inc.	Compute

Then click Create



	PUBLISHER	CATEGORY
Logic App	Microsoft	Web + Mobile
Sumo Logic for Azure Web Apps	Sumo Logic	Compute
Sumo Logic for Azure Audit Logs	Sumo Logic	Compute
On-premises data gateway (preview)	Microsoft	Web + Mobile
Integration Account	Microsoft	Web + Mobile
SQL Database	Microsoft	Databases
Realm Mobile Platform Developer Edition	Realm	Compute
Mendix Pro Edition - 250 users	Mendix Inc.	Compute
Mendix Pro Edition - 100 users	Mendix Inc.	Compute
Mendix Pro Edition - 50 users	Mendix Inc.	Compute
Mendix Pro Edition - BYOL	Mendix Inc.	Compute

Logic Apps allow developers to design workflows that articulate intent via a trigger and series of steps, each invoking an App Service API app whilst securely taking care of authentication and best practices like durable execution.

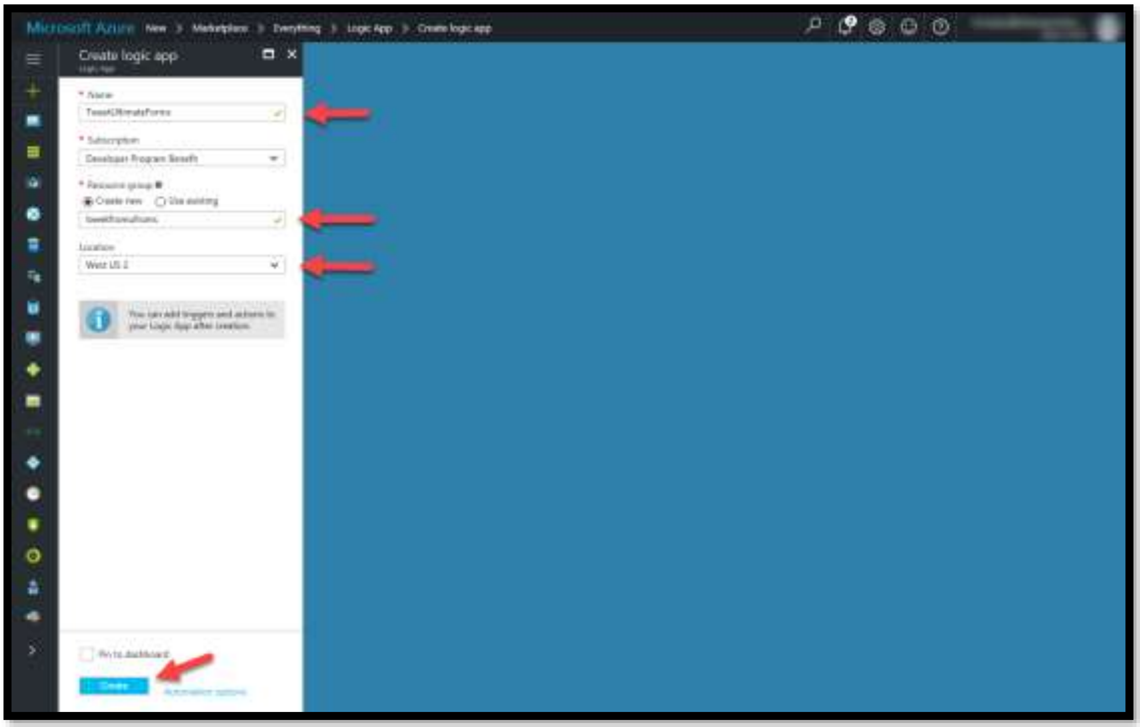
Easy to use design tools - Logic Apps can be designed end-to-end in the browser. Start with a trigger - from a simple schedule to whenever a tweet appears about your company. Then orchestrate any number of actions using the rich gallery of connectors.

Compose SaaS easily - Even composition tasks that are easy to describe are difficult to implement in code. Logic Apps make it a cinch to connect disparate systems. Want to create a task in CRM based on activity on your Facebook or Twitter accounts? Want to connect your cloud marketing solution to your on-premises billing system? Logic apps are the fastest, most reliable way to deliver solutions to these problems.

Extensibility baked in - Don't see the connector you need? Logic Apps are part of the App Service suite and designed to work with API apps; you can easily create your own API app to use as a connector. Build a new app just for you, or share and monetize in the marketplace.

Real integration horsepower - Start easy and grow as you need. Logic Apps can easily leverage the power of BizTalk, Microsoft's industry leading integration solution to enable integration professionals to build the solutions they need.

Give your Logic App a name, choose a subscription and then create a new resource group in the region of your choice and click create



Microsoft Azure
New > Marketplace > Everything > Logic App > Create logic app
Create logic app
Logic App

* Name
TweetUltimateForms

* Subscription
Developer Program Benefit

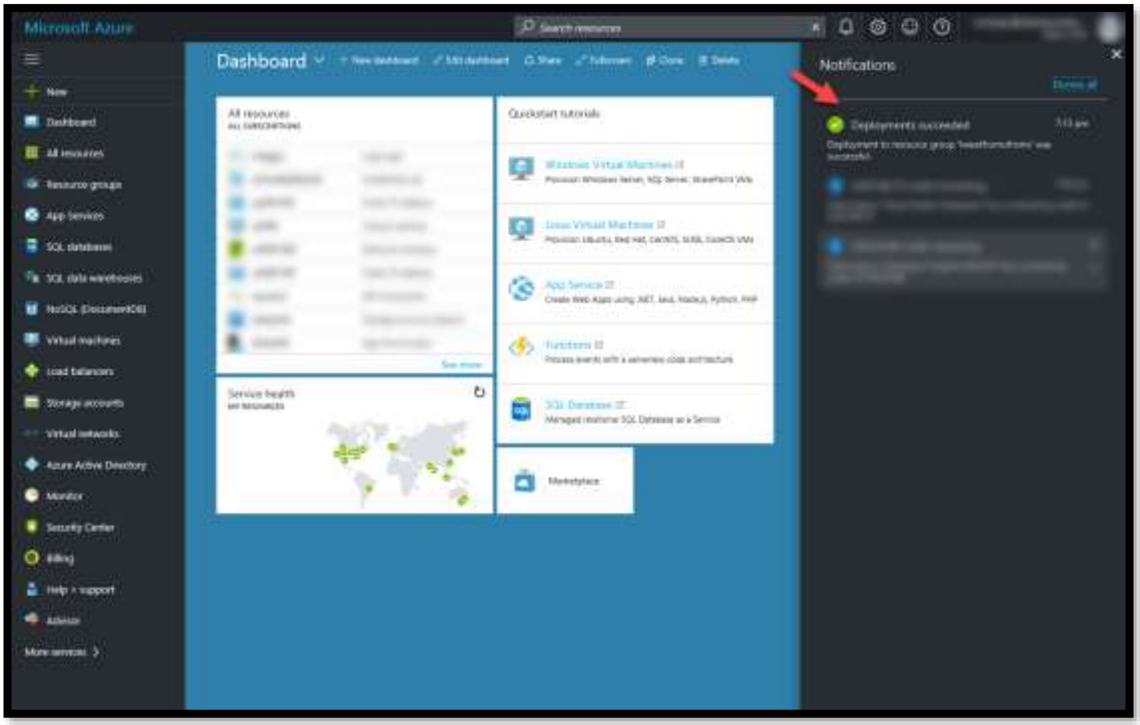
* Resource group
● Create new ○ Use existing
tweetforesolutions

Location
West US 2

You can add triggers and actions to your Logic App after creation.

□ Pin to dashboard

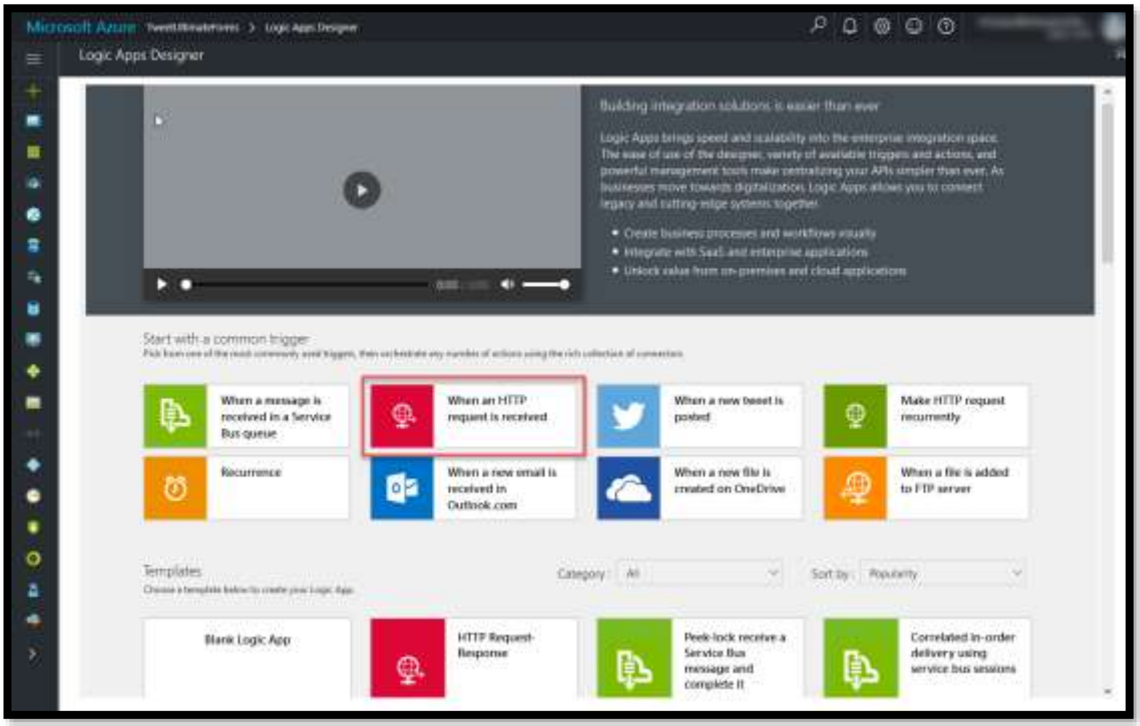
You should then be taken back to the portal home page and soon afterwards get a notification that the logic app was created, you can click the notification to be taken to the logic app design surface.



Deployments succeeded 7:13 am

Deployment to resource group 'tweetfromutrons' was successful.

Choose the “When an HTTP Request is Received” Trigger when prompted



Building integration solutions is easier than ever

Logic Apps brings speed and scalability into the enterprise integration space. The ease of use of the designer, variety of available triggers and actions, and powerful management tools make centralizing your APIs simpler than ever. As businesses move towards digitalization, Logic Apps allows you to connect legacy and cutting-edge systems together.

- Create business processes and workflows visually
- Integrate with SaaS and enterprise applications
- Unlock value from on-premises and cloud applications

Start with a common trigger

Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich collection of connectors

When a message is received in a Service Bus queue
When an HTTP request is received
When a new tweet is posted
Make HTTP request recurrently
Recurrence
When a new email is received in Outlook.com
When a new file is created on OneDrive
When a file is added to FTP server

Templates

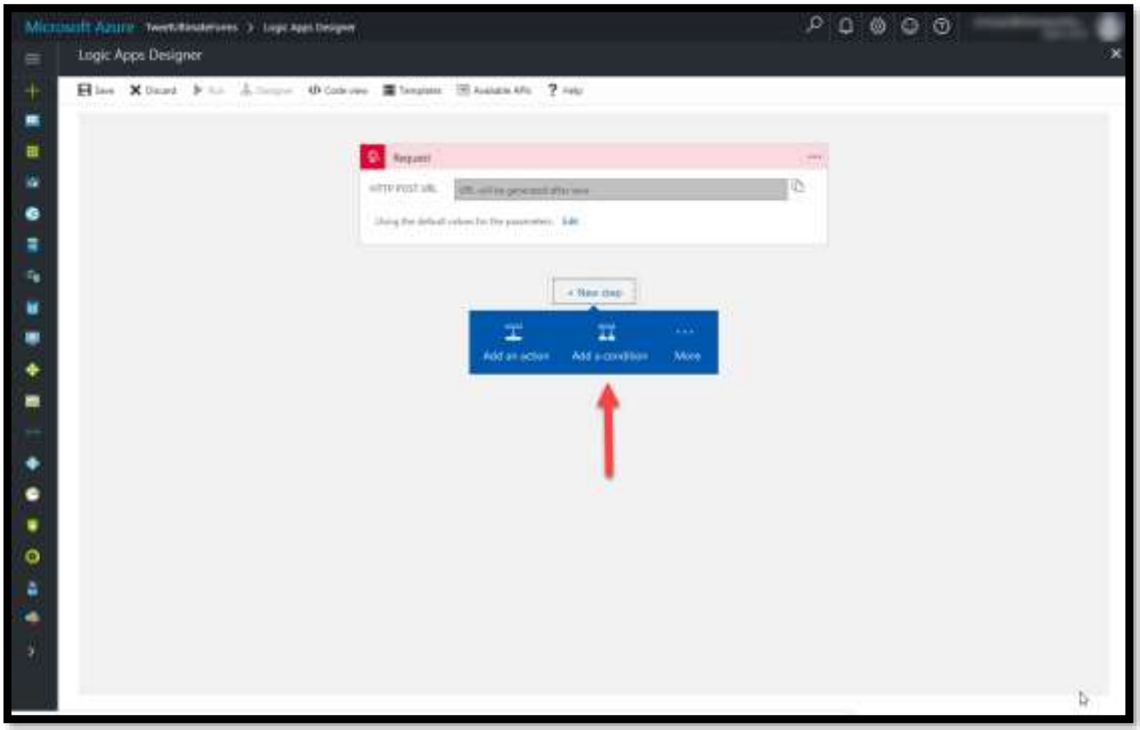
Choose a template below to create your Logic App

Category : All

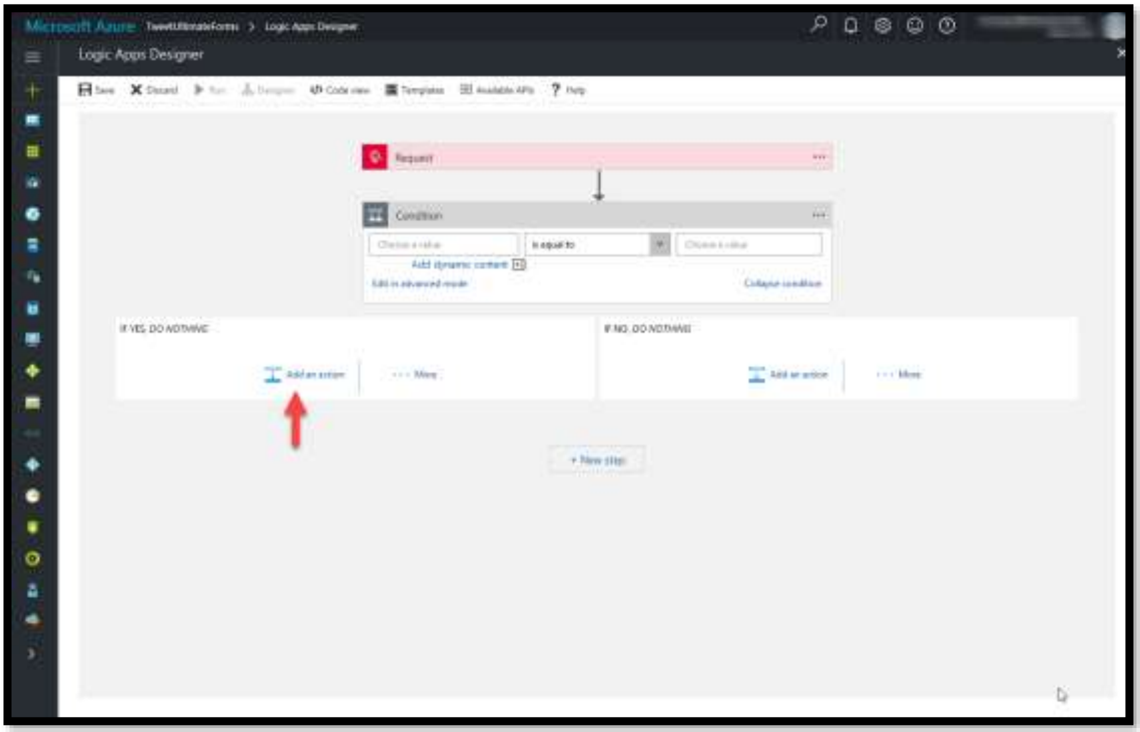
Sort by : Popularity

Blank Logic App
HTTP Request-Response
Peek-lock receive a Service Bus message and complete it
Correlated in-order delivery using service bus sessions

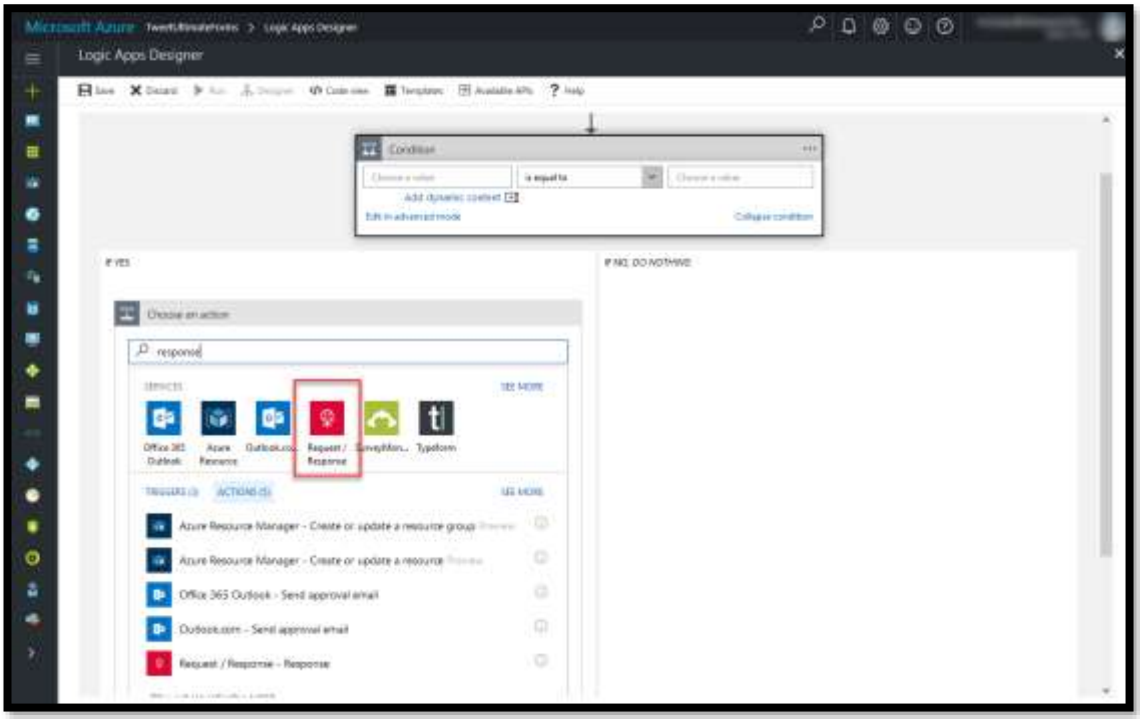
Once the design surface appears, choose the option to add a condition. This is the area where we will specify our super secret authentication phrase so no one else can tweet on our behalf.



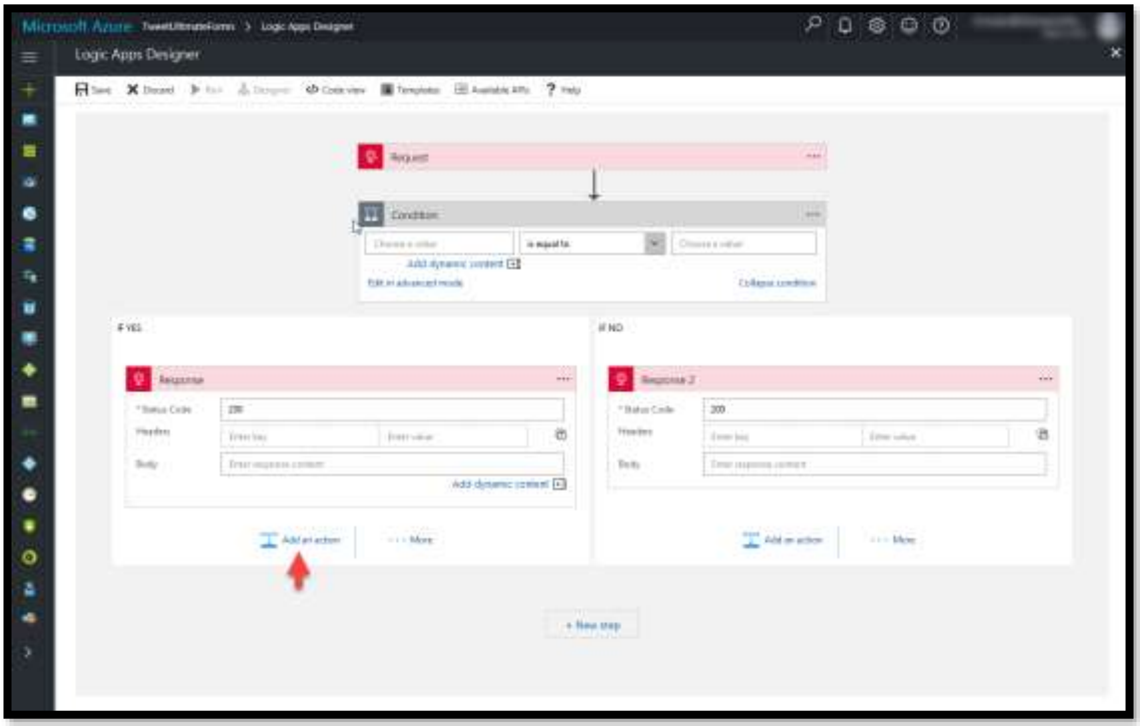
Don’t configure anything yet on the surface, instead on each the yes and the now sides of the condition, add a “response” action.



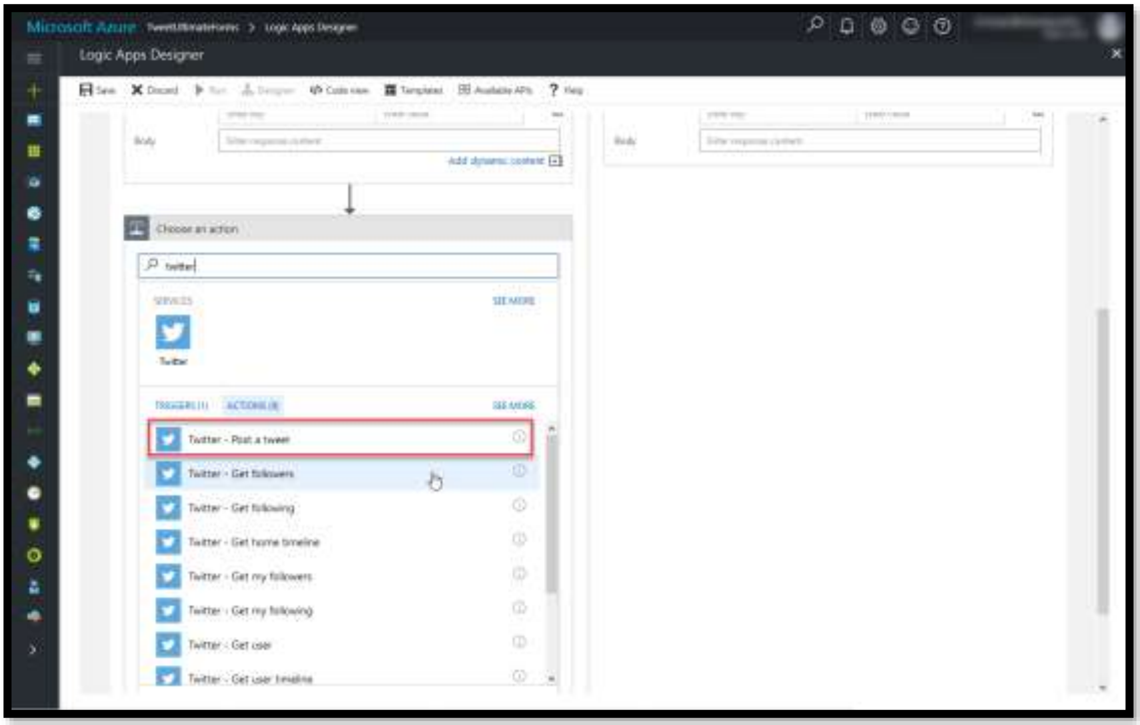
Type in “Response” and then select the appropriate item when it comes up in the search results



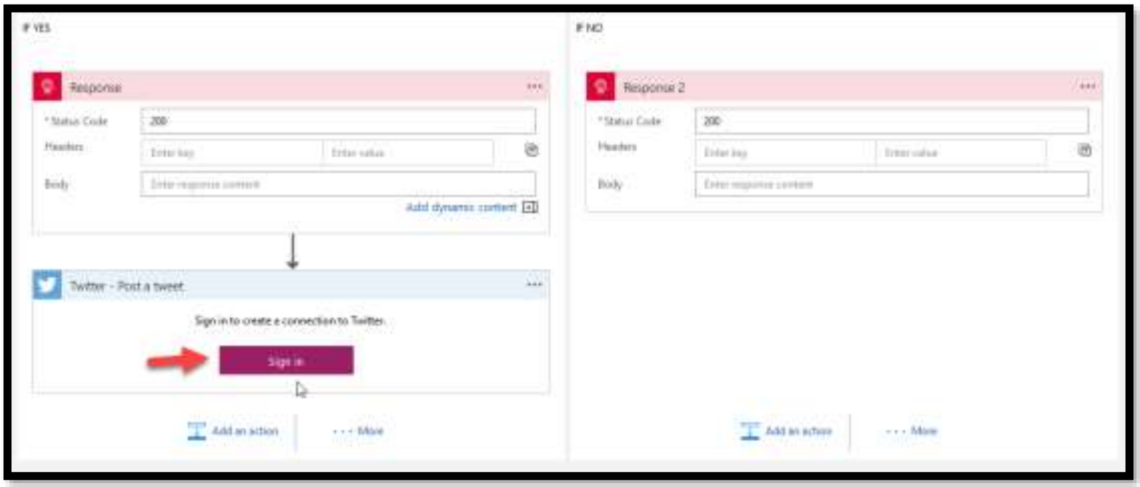
Once you have the responses on both sides of the condition, add another action to the “yes” side.



On the Yes side, we want to place a twitter action. So in the text box search for twitter. In the results box, choose Twitter – Post a Tweet

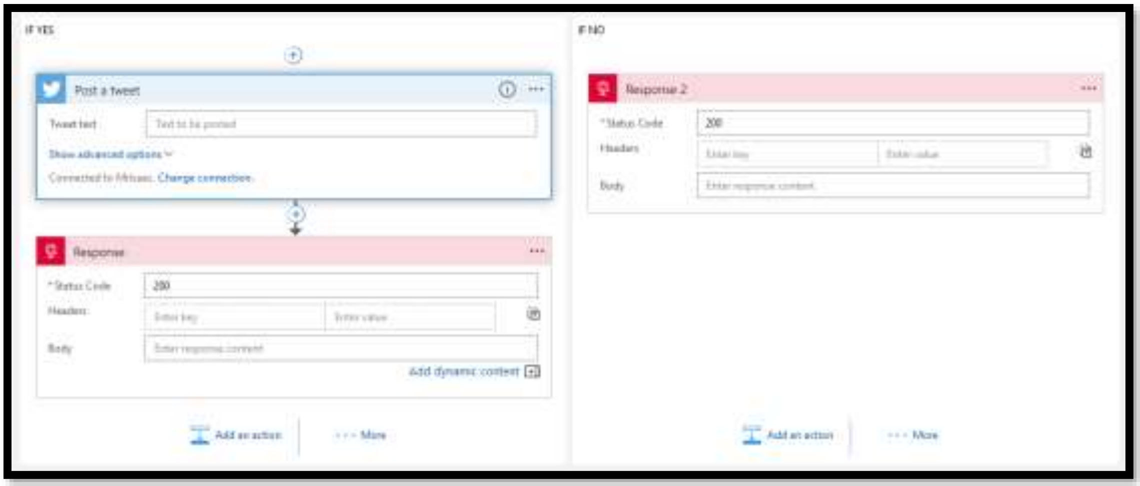


Click Sign In to twitter and follow the normal twitter authentication steps



Sign in to create a connection to Twitter.

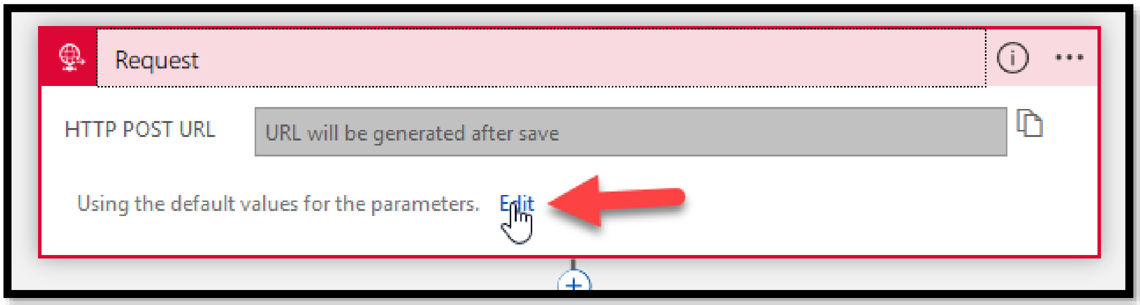
Once that is done, drag the twitter action before the Response action so it looks like below:



Response: * Status Code 200

Response 2: * Status Code 200

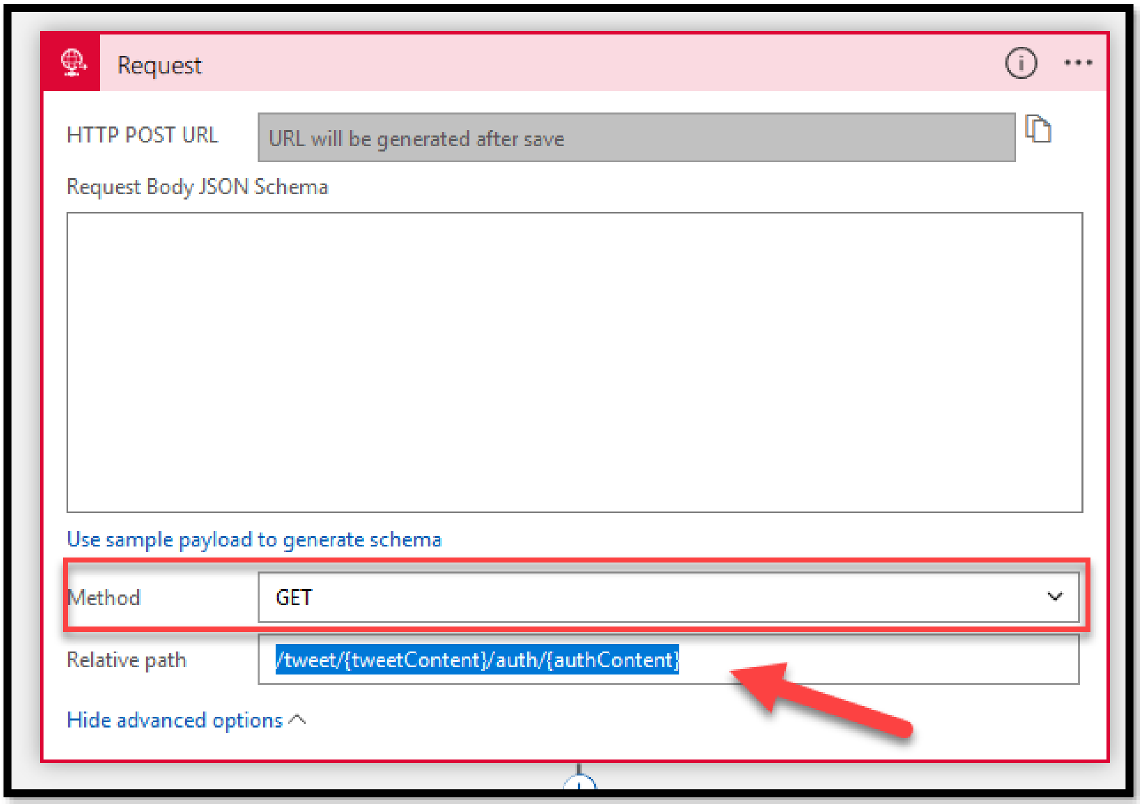
At this point we need to configure the actual actions and triggers so we can actually post the tweet. So scroll back up to the Request trigger and expand it by clicking on it.



HTTP POST URL: URL will be generated after save

Using the default values for the parameters. Edit

Click edit to show the properties for the trigger


Request
HTTP POST URL
URL will be generated after save
Request Body JSON Schema
Use sample payload to generate schema
Hide advanced options

Method: GET

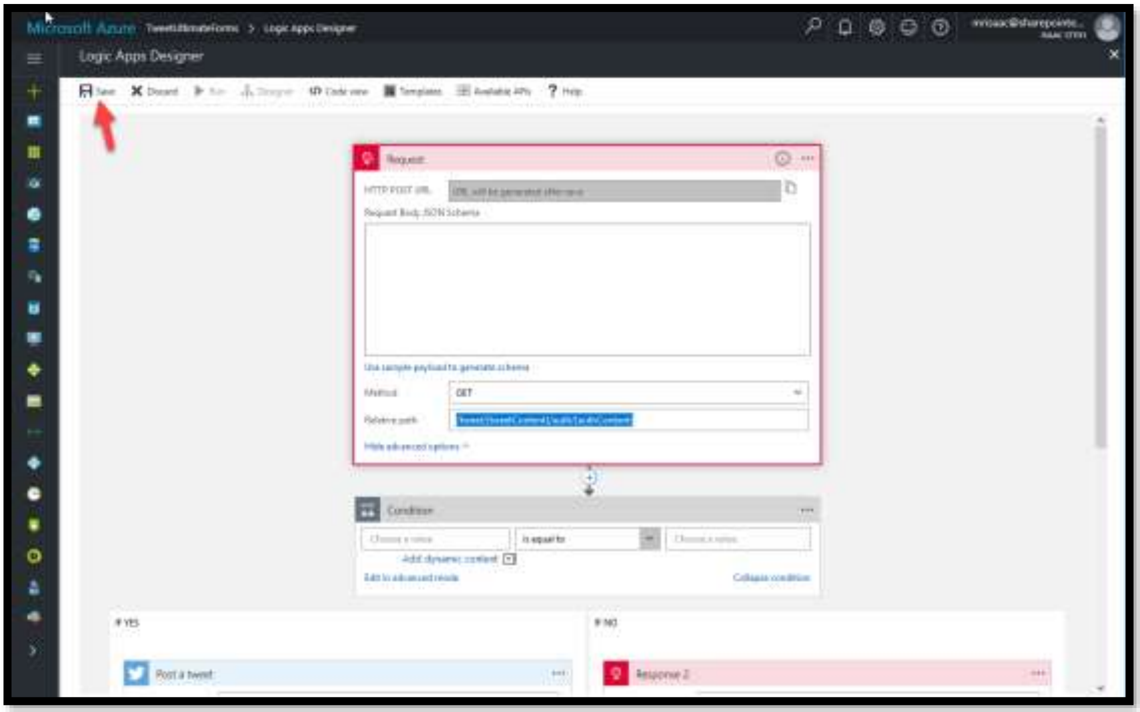
Relative path: /tweet/{tweetContent}/auth/{authContent}

Change the Method to Get

And then insert this into the Relative Path

/tweet/{tweetContent}/auth/{authContent}

At this point you need to click Save on the menu ribbon for the changes to persist and the form to update.



HTTP POST URL: URL will be generated after save

Request Body JSON Schema

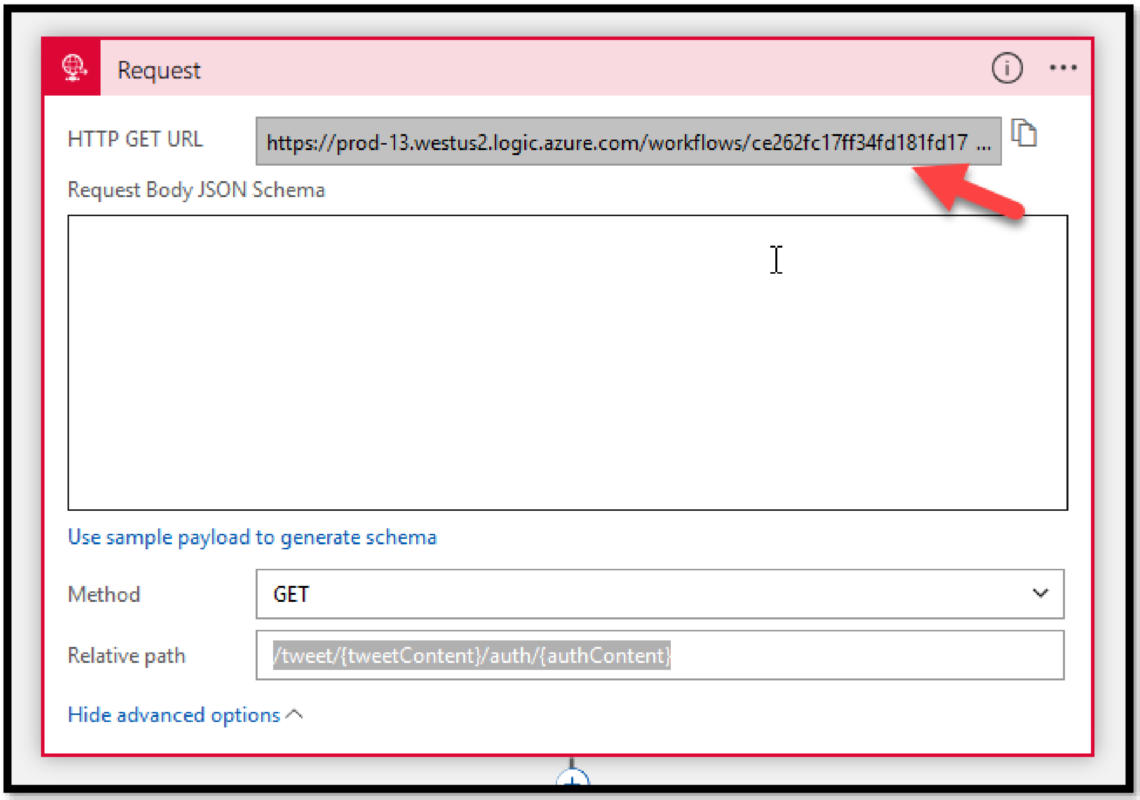
Use sample payload to generate schema

Method: GET

Relative path: /tweet/{tweetContent}/auth/{authContent}

Hide advanced options

You will notice that the URL Specified is now populated after the save and looks something like this



HTTP GET URL: https://prod-13.westus2.logic.azure.com/workflows/ce262fc17ff34fd181fd17 ...

Request Body JSON Schema

Use sample payload to generate schema

Method: GET

Relative path: /tweet/{tweetContent}/auth/{authContent}

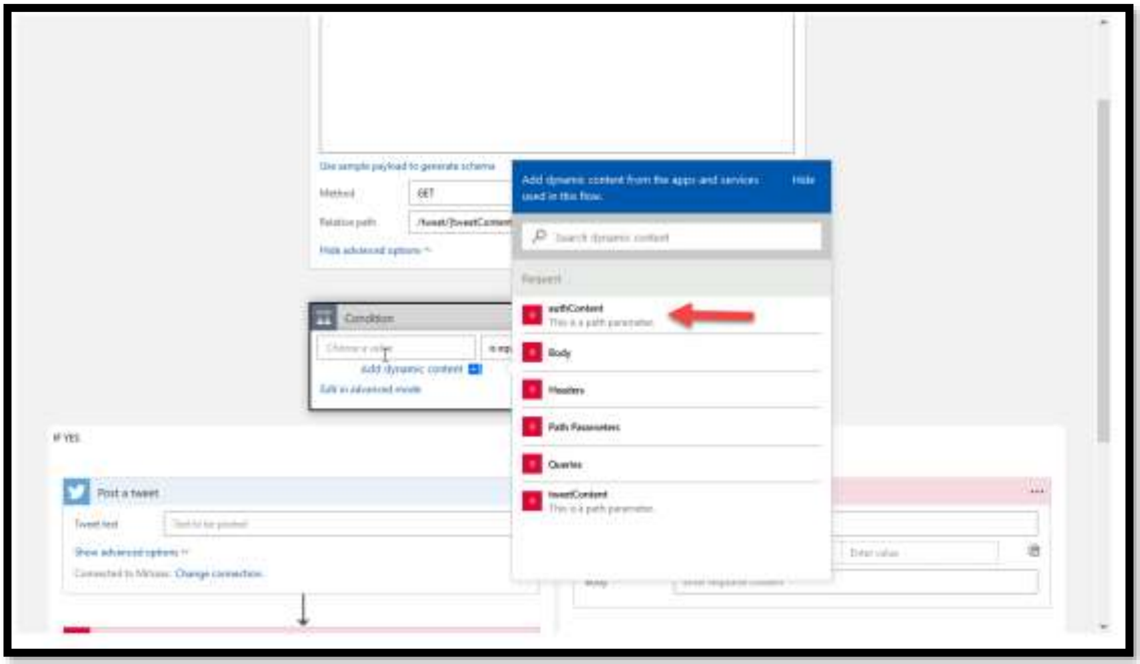
Hide advanced options

Though the URL change depending on the environment that you specified in the beginning steps.

For this example, the URL Generated was:

> https://prod-13.westus2.logic.azure.com/workflows/ce262fc17ff34fd181fd172f9e562779/triggers/manual/paths/invoke/tweet/{tweetContent}/auth/{authContent}?api-version=2016-06-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=flysWTtmJcPlvVyqg330A__LU_OypMSDwVVS18_27Vs

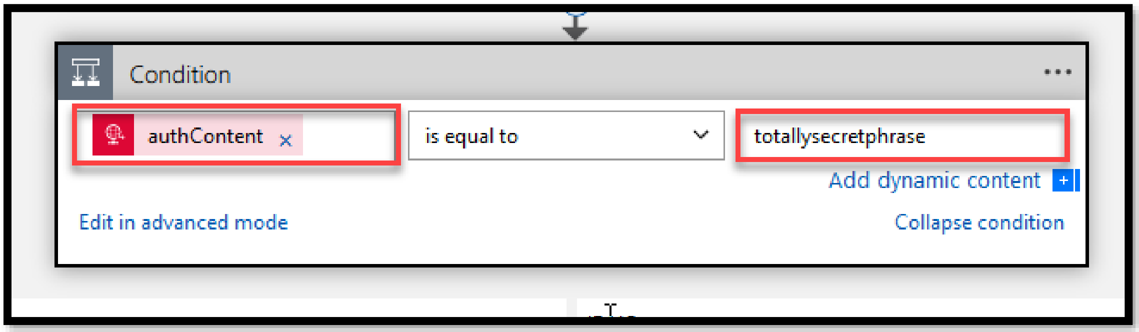
Now scroll down slightly to the condition and add click in the choose a value box



Method: GET

Relative path: /tweet/{tweetContent

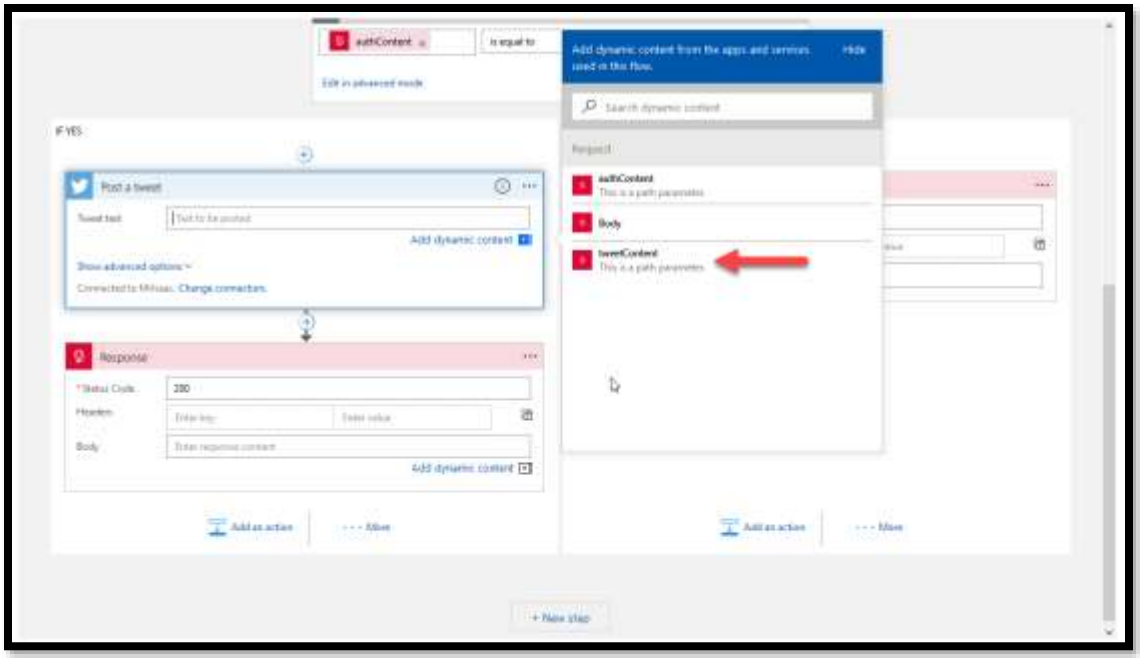
Now that we have configured the relative parameters above, we can use them in our Logic Flow, so choose authContent and in the value box, enter in a super secret key phrase.



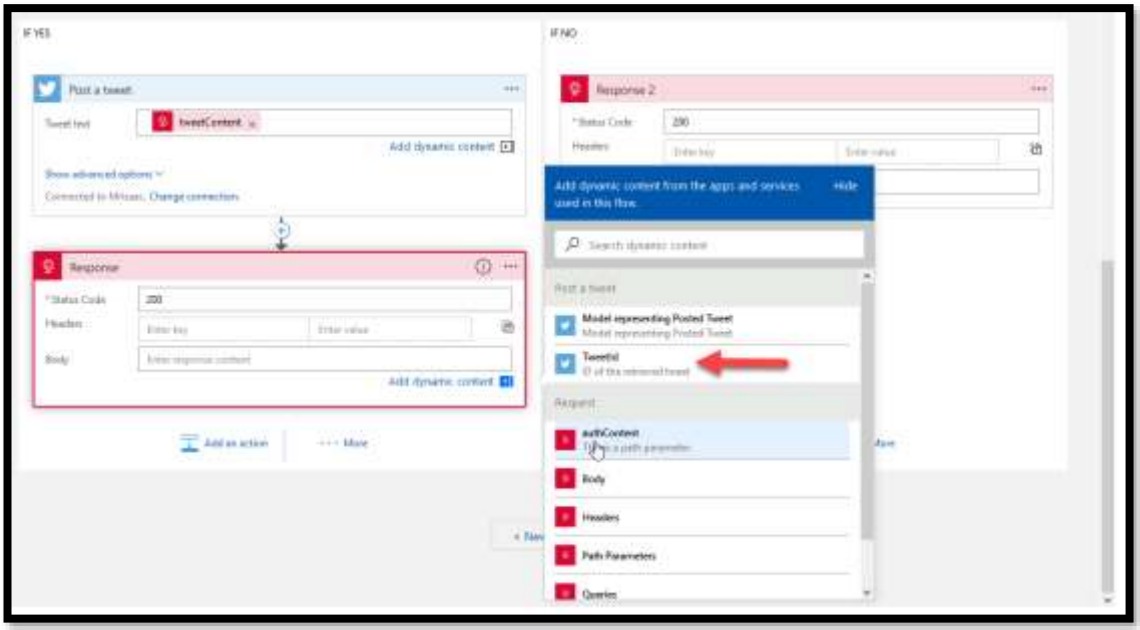
Condition

authContent is equal to totallysecretphrase

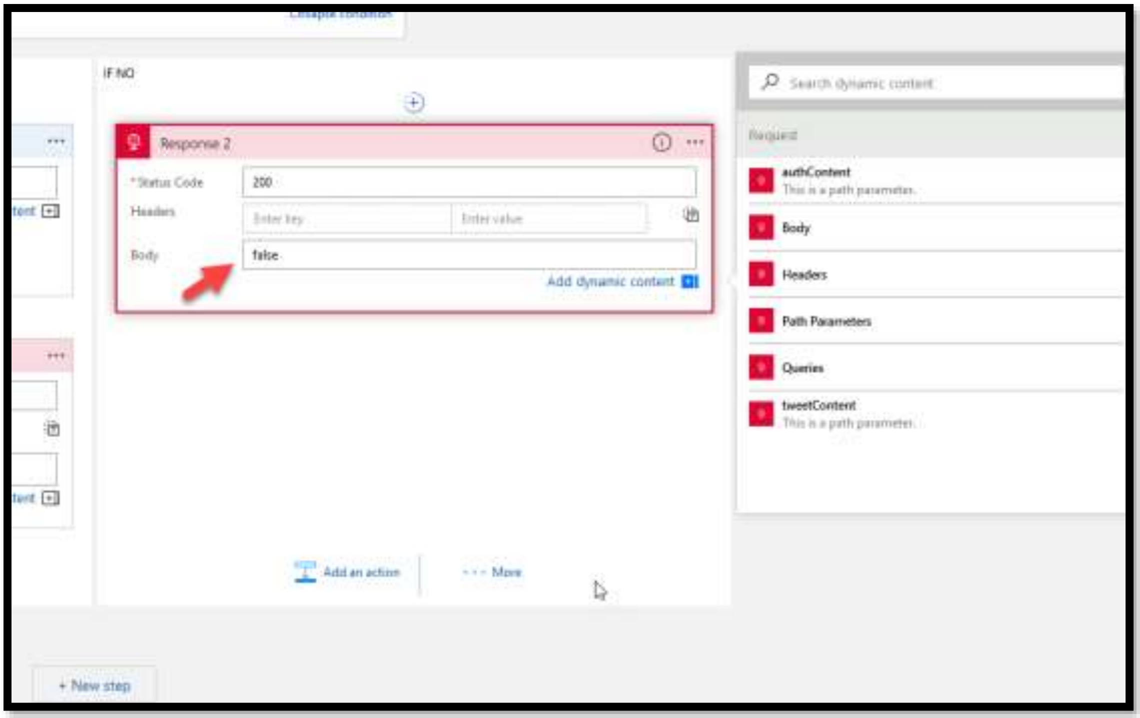
Then move down to the Twitter action and click in the Text to be posted box.



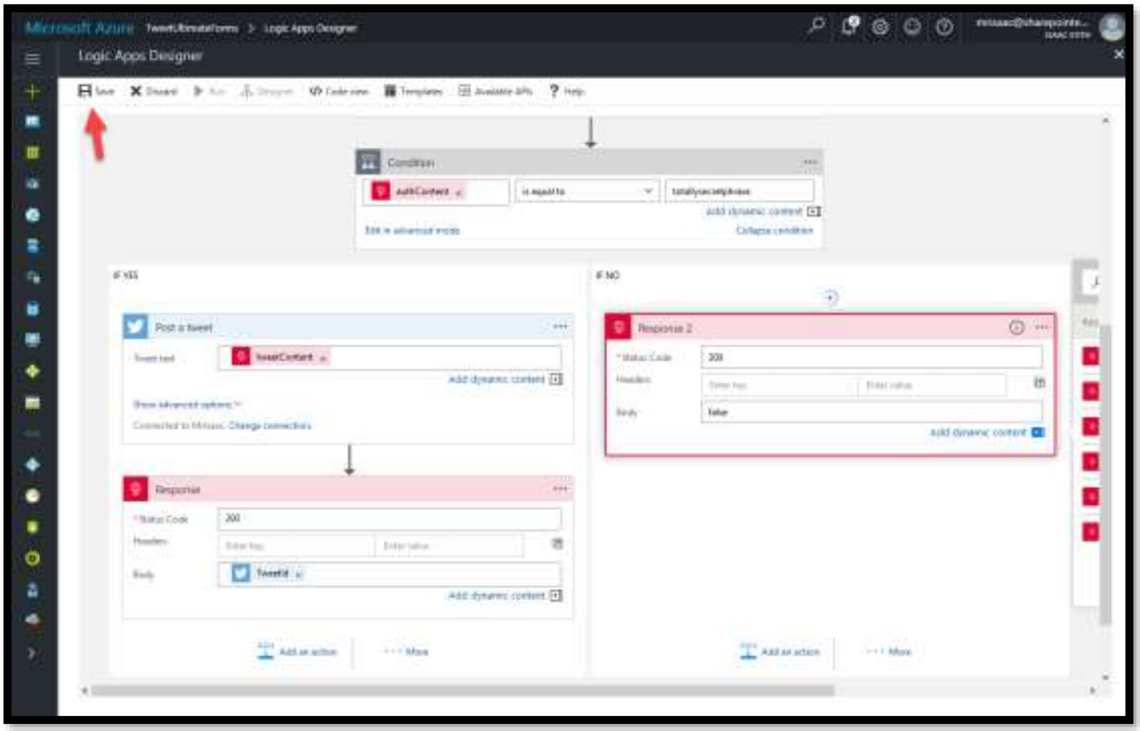
Choose the tweetContent option and then in the Response Dialog below, click in the body text box.



Choose the TweedID returned from the previous action as the body content.
Over on the No side, we need to specify a value as well. So go to the Body text box there and enter in “false”



Now that we have configured all the actions and triggers, make sure you click the save button.



And we are done with the Azure Logic App portion of the form action.

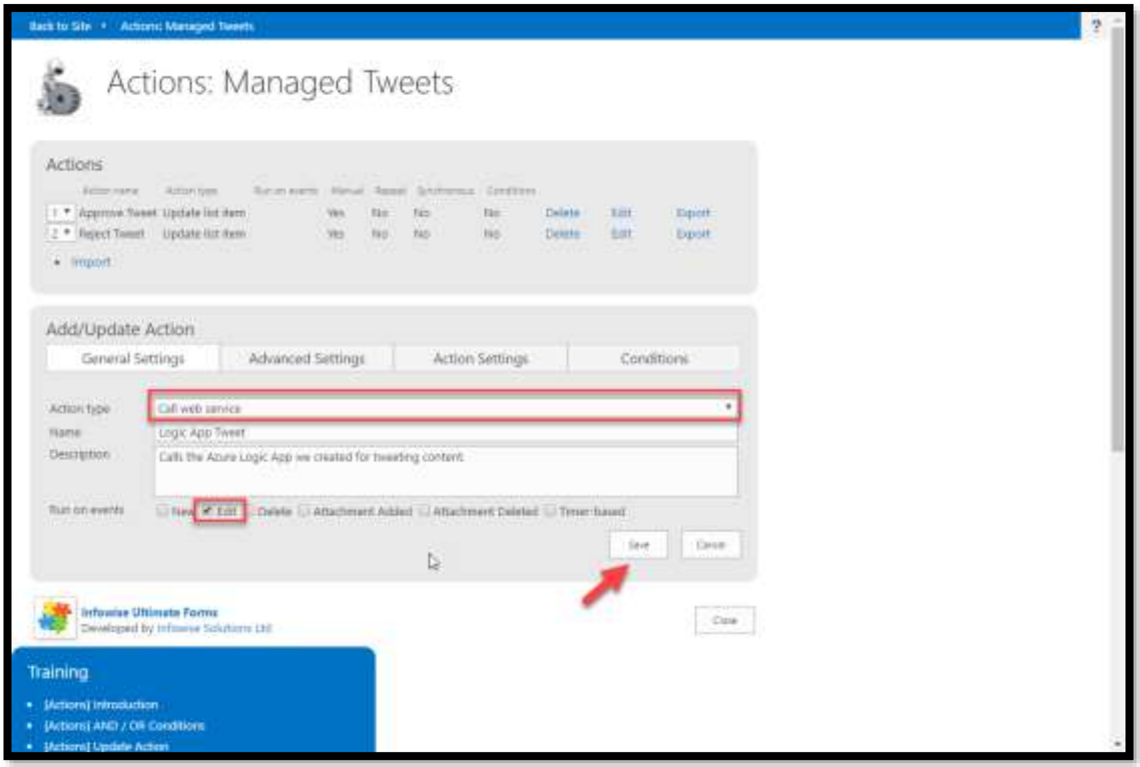
Now you have a URL that accepts inputs for tweets with a specific super-secret string. So, if we wanted to send a tweet with the content “I love Ultimate Forms” we would simply call the URL below, see the bolded sections.

https://prod-13.westus2.logic.azure.com/workflows/ce262fc17ff34fd181fd172f9e562779/triggers/manual/paths/invoke/**tweet/I%20Love%20Ultimate%20Forms/auth/totallysecretphrase?**api-version=2016-06-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=fIysWTtmJcPlvVyqg330A__LU_OypMSDwVVS18_27Vs

Call the Custom Action

At this point we have an approval action and we have a web service that accepts inputs for tweets. All we need to do is to tie the two pieces together. So, go back to your list design in Infowise and choose actions.

On the first page choose “Call a web service” from the action type list



Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

Action type: Call web service

Name: Logic App Tweet

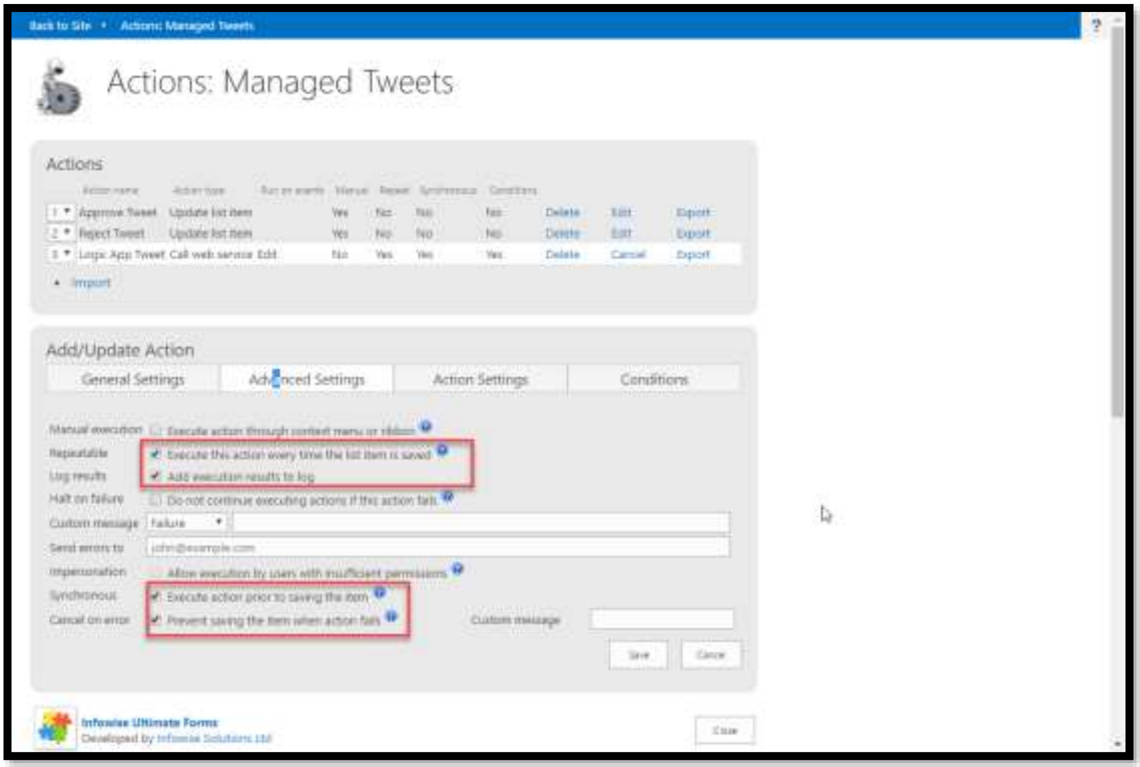
Description: Calls the Azure Logic App we created for tweeting content

Run on events: □ New ☑ Edit □ Delete □ Attachment Added □ Attachment Deleted □ Timer-based

Training

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action

On the advanced settings, make sure you have synchronous and logging checked



Back to Site • Actions: Managed Tweets

Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
3	Logic App Tweet	Call web service	Edit	No	Yes	Yes	Yes	Delete	Cancel	Export

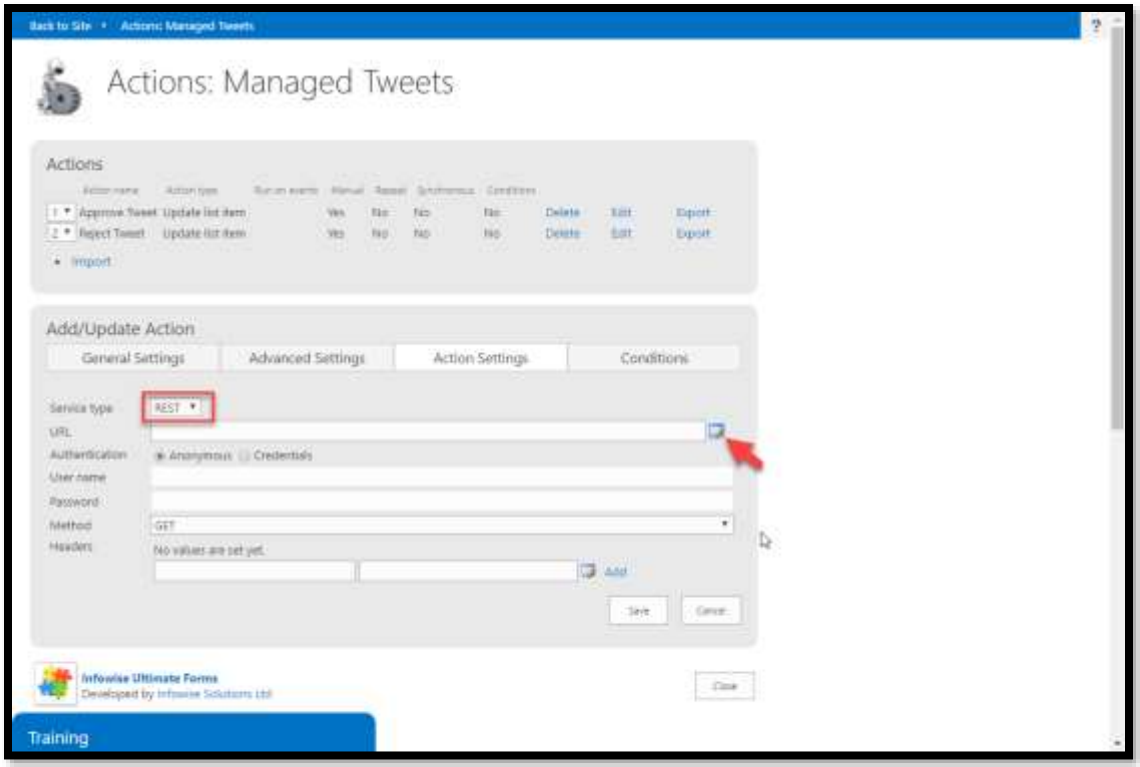
Import

Add/Update Action

General Settings | Advanced Settings | Action Settings | Conditions

Manual execution □ Execute action through context menu or ribbon
Repeatable ☑ Execute this action every time the list item is saved
Log results ☑ Add execution results to log
Halt on failure □ Do not continue executing actions if this action fails
Custom message Failure
Send errors to john@example.com
Impersonation □ Allow execution by users with insufficient permissions
Synchronous ☑ Execute action prior to saving the item
Cancel on error ☑ Prevent saving the item when action fails
Custom message

On the action settings tab, make sure you choose REST and then click the values dialog to input the URL



Back to Site • Actions: Managed Tweets

Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

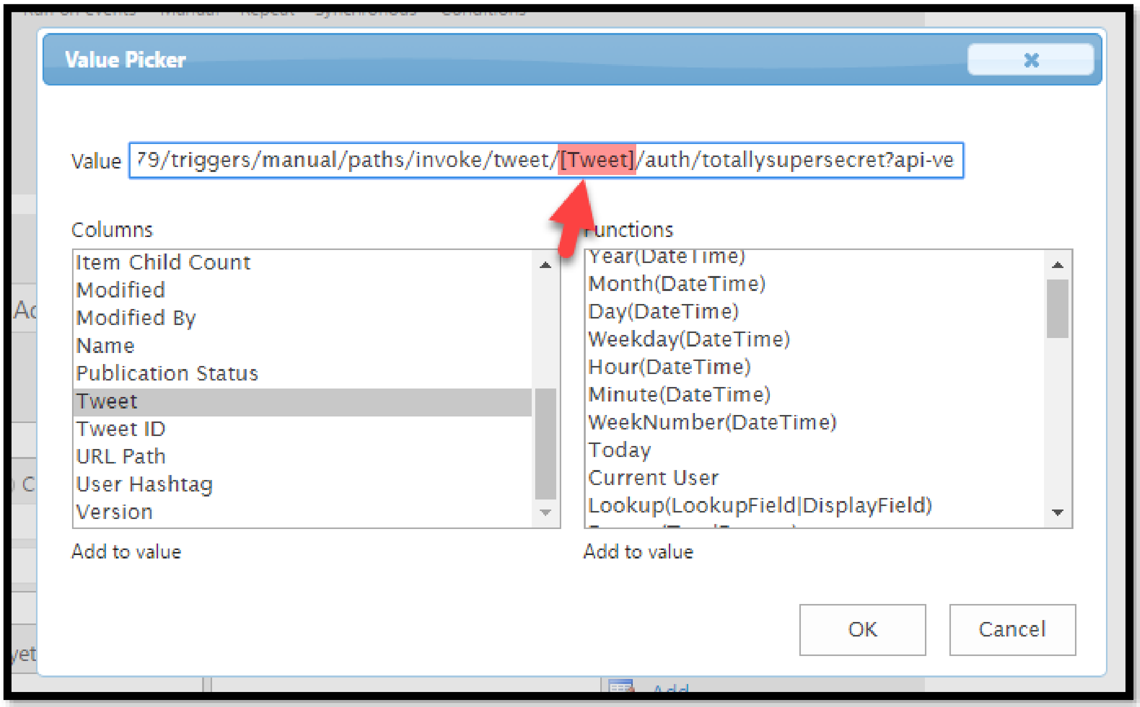
Import

Add/Update Action

General Settings | Advanced Settings | Action Settings | Conditions

Service type REST
URL
Authentication ● Anonymous ○ Credentials
User name
Password
Method GET
Headers No values are set yet.

Copy the URL we got from saving the Logic App in the HTTP Trigger in the first step, and paste it into this field.

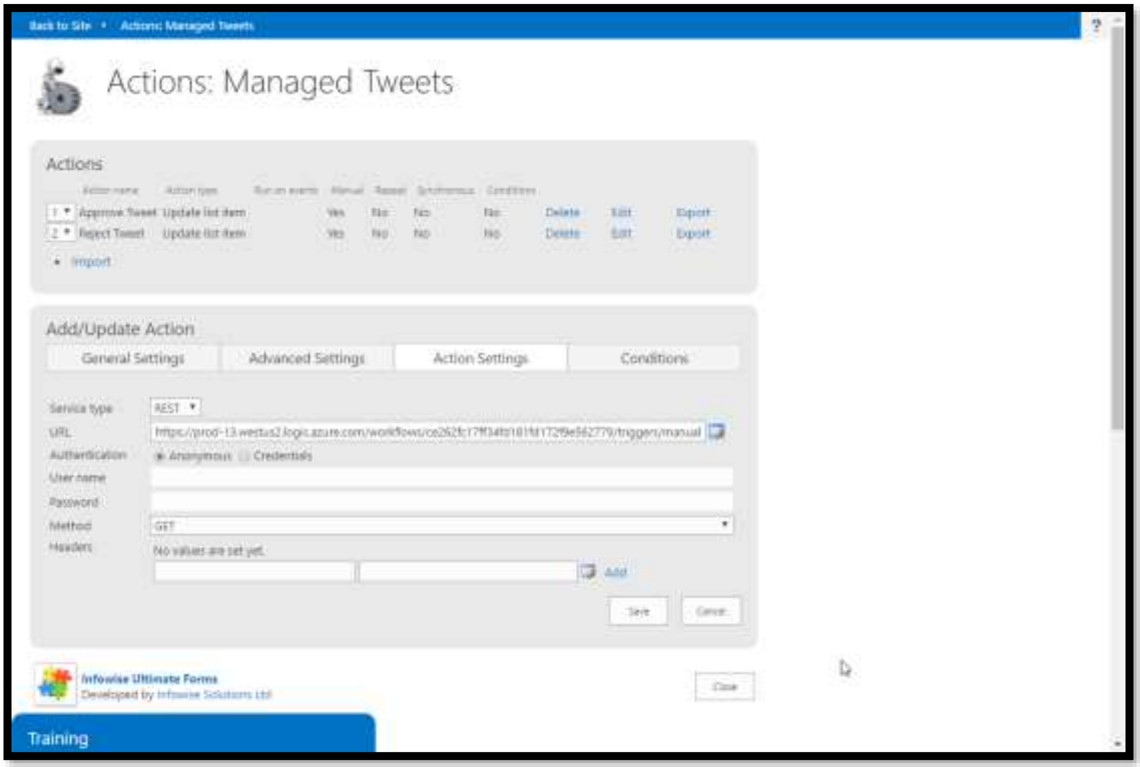


Value Picker

Value 79/triggers/manual/paths/invoke/tweet/[Tweet]/auth/totallysupersecret?api-ve

Make sure you replace the {tweetContent} token with [Tweet] to insert the value of the list item in the call. And you also need to replace {authContent} with whatever super secret phrase you chose.

Click Okay and verify everything looks correct.



Back to Site • Actions: Managed Tweets

Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export

- Import

Add/Update Action

General Settings | Advanced Settings | Action Settings | Conditions

Service type: REST ▾

URL: https://prod-13.westus2.logic.azure.com/workflows/ce262fc17ff34fb181fd172f9e562779/triggers/manual

Authentication: ● Anonymous ○ Credentials

User name:

Password:

Method: GET ▾

Headers: No values are set yet.

Add

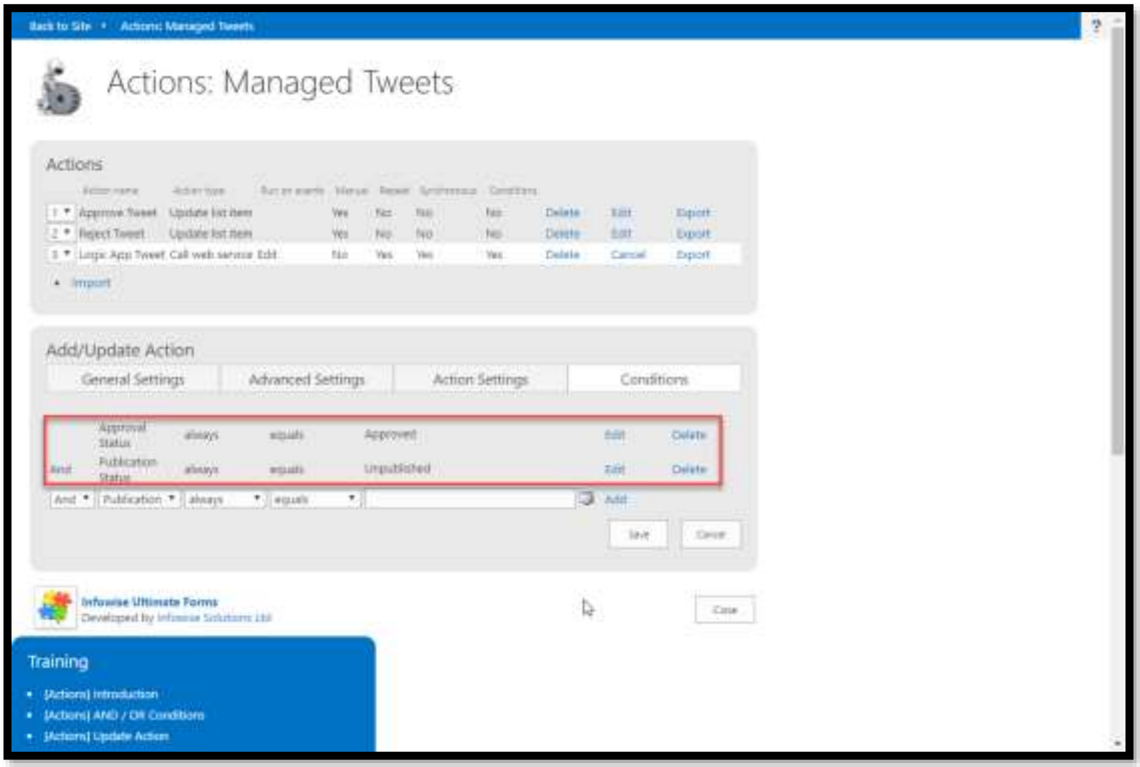
Save | Cancel

Infowise Ultimate Forms
Developed by Infowise Solutions Ltd

Close

Training

One last step is to add conditions so that anything that hasn't been approved yet, or has failed publishing is not tweeted.



Back to Site • Actions: Managed Tweets

Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
3 ▾	Logic App Tweet	Call web service Edit		No	Yes	Yes	Yes	Delete	Cancel	Export

- Import

Add/Update Action

General Settings | Advanced Settings | Action Settings | Conditions

	Approval Status	always	equals	Approved	Edit	Delete
And	Publication Status	always	equals	Unpublished	Edit	Delete

And ▾ | Publication ▾ | always ▾ | equals ▾ | Add

Save | Cancel

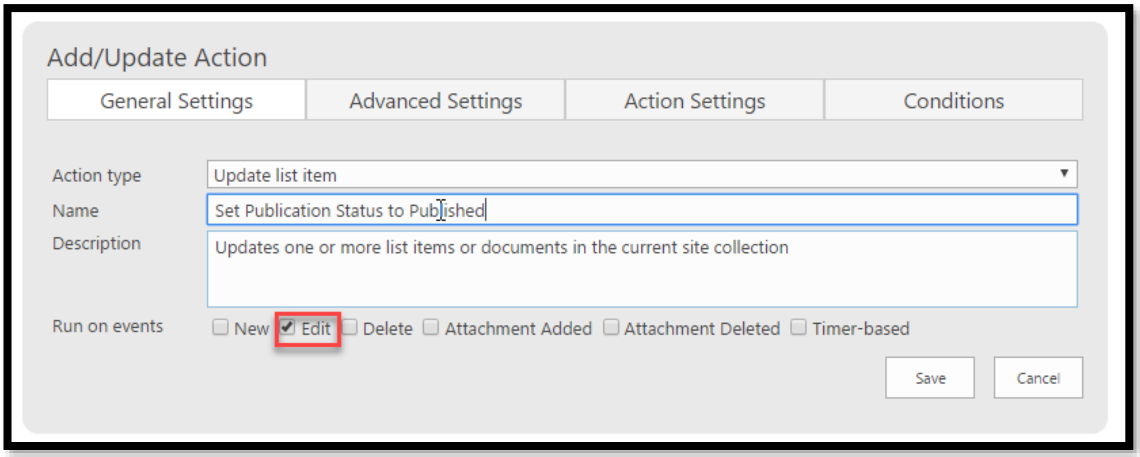
Infowise Ultimate Forms
Developed by Infowise Solutions Ltd

Close

Training

- [Actions] Introduction
- [Actions] AND / OR Conditions
- [Actions] Update Action

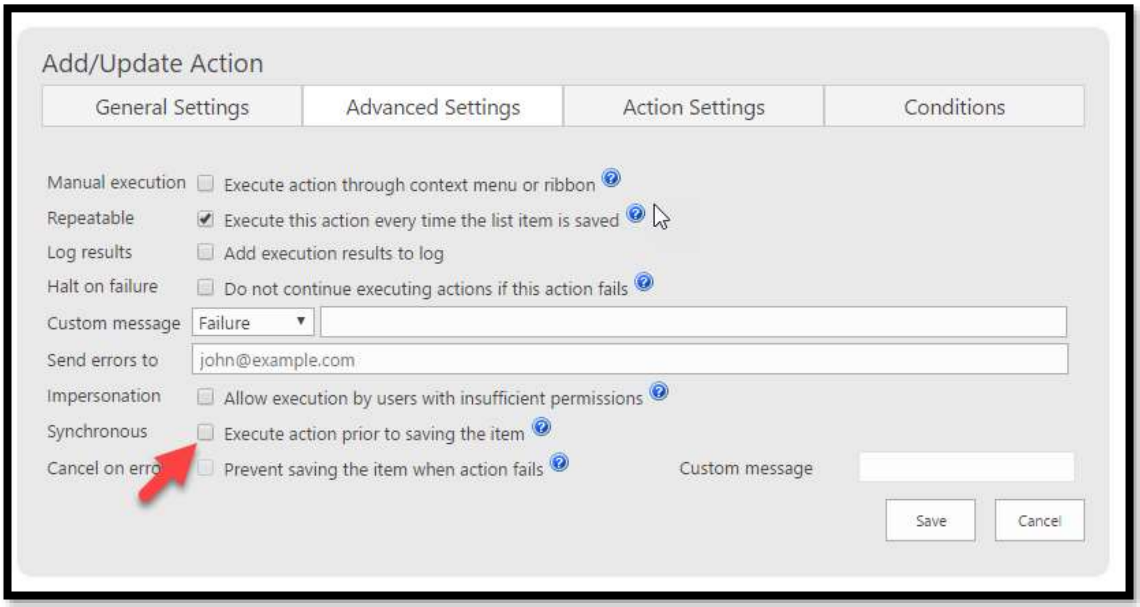
Then after we save this action, we need to create another action which updates the status to published if the tweet is sent successfully



Add/Update Action

Action type	Update list item
Name	Set Publication Status to Published
Description	Updates one or more list items or documents in the current site collection
Run on events	□ New ☑ Edit □ Delete □ Attachment Added □ Attachment Deleted □ Timer-based

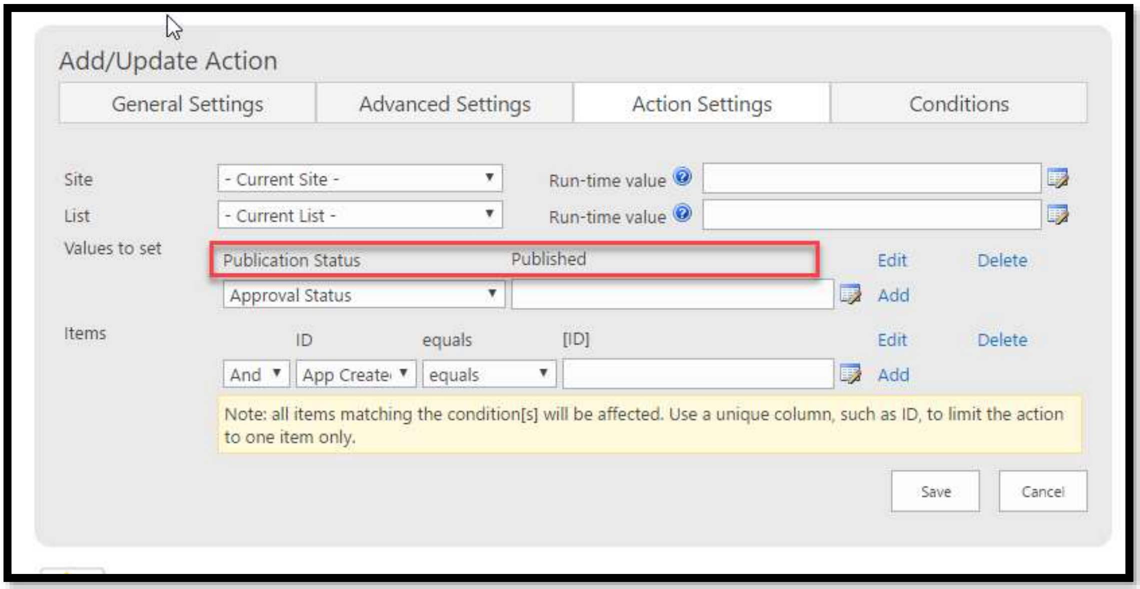
So create an update list item action that runs on Edits



Add/Update Action

Manual execution	□ Execute action through context menu or ribbon
Repeatable	☑ Execute this action every time the list item is saved
Log results	□ Add execution results to log
Halt on failure	□ Do not continue executing actions if this action fails
Custom message	Failure
Send errors to	john@example.com
Impersonation	□ Allow execution by users with insufficient permissions
Synchronous	□ Execute action prior to saving the item
Cancel on erro	□ Prevent saving the item when action fails Custom message

Make sure synchronous is unchecked, we want this to happen after the web service action



Site: - Current Site - | Run-time value

List: - Current List - | Run-time value

Values to set: Publication Status — Published (Edit, Delete)

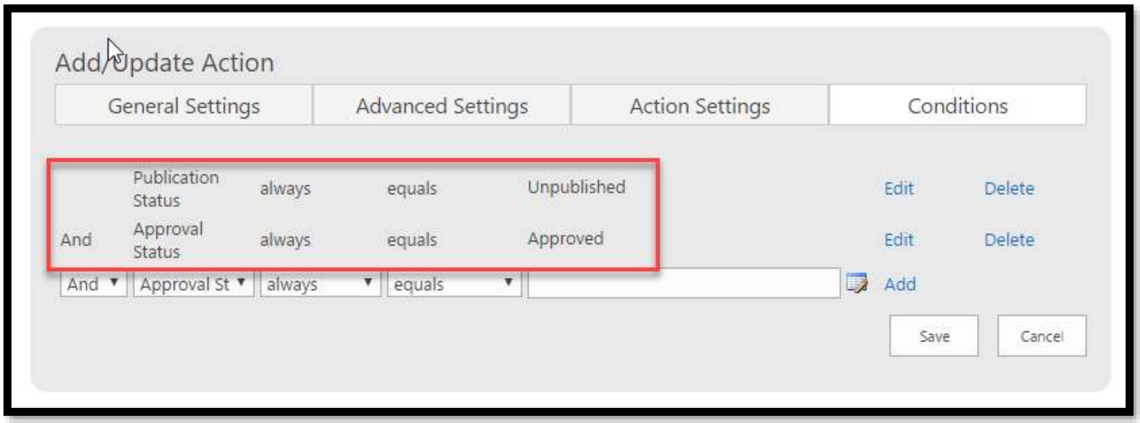
Approval Status (Add)

Items: ID equals [ID] (Edit, Delete)

And | App Create | equals (Add)

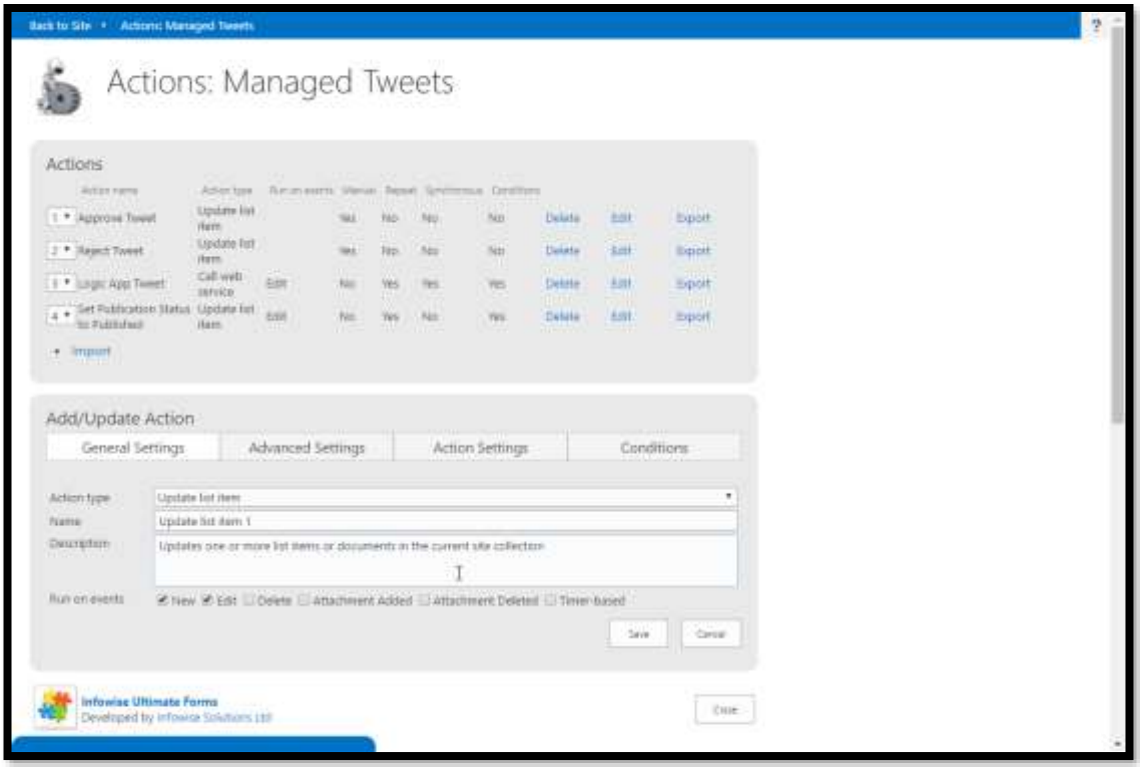
Note: all items matching the condition[s] will be affected. Use a unique column, such as ID, to limit the action to one item only.

Update the publication status to Published



	Publication Status	always	equals	Unpublished	Edit	Delete
And	Approval Status	always	equals	Approved	Edit	Delete
And	Approval St	always	equals			Add

And also set the same conditions as the web service action and click save.



Actions: Managed Tweets

Actions

	Action name	Action type	Run on events	Manual	Repeat	Synchronous	Conditions			
1 ▾	Approve Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
2 ▾	Reject Tweet	Update list item		Yes	No	No	No	Delete	Edit	Export
3 ▾	Logic App Tweet	Call web service	Edit	No	Yes	Yes	Yes	Delete	Edit	Export
4 ▾	Set Publication Status to Published	Update list item	Edit	No	Yes	No	Yes	Delete	Edit	Export

Add/Update Action

General Settings | Advanced Settings | Action Settings | Conditions

Action type: Update list item ▾

Name: Update list item 1

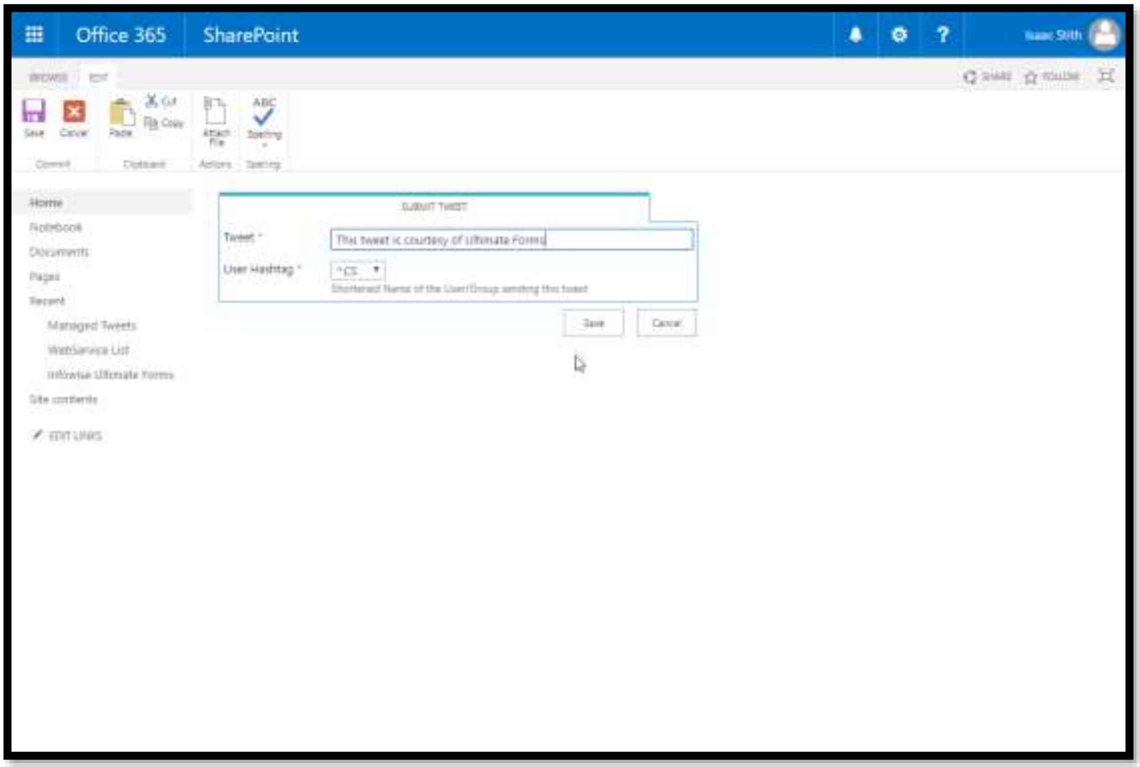
Description: Updates one or more list items or documents in the current site collection

Run on events: ☑ New ☑ Edit □ Delete □ Attachment Added □ Attachment Deleted □ Timer-based

So we should now have a form that publishes a tweet when its approved

Running the Process

Now that we have all the pieces in place and tied together, lets run the form to see if we actually get a tweet. Go back to your SharePoint list and add a new Managed Tweet.



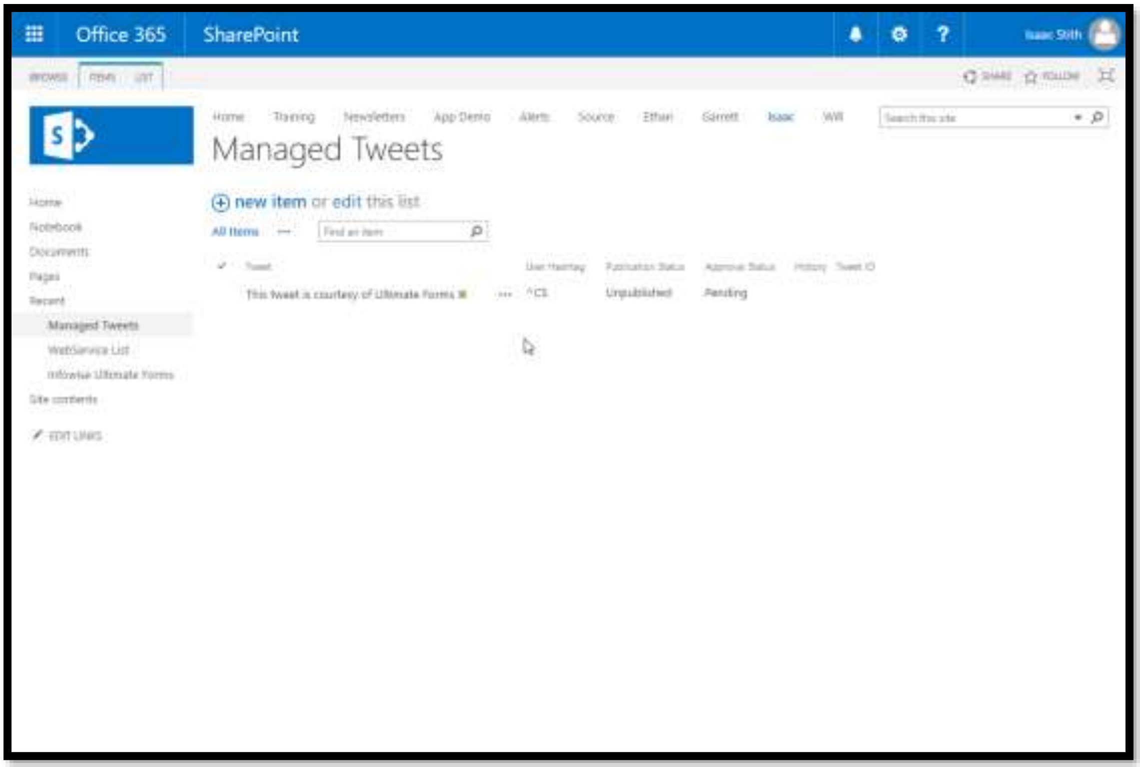
SUBMIT TWEET

Tweet *: This tweet is courtesy of Ultimate Forms

User Hashtag *: ^CS

Shortened Name of the User/Group sending this tweet

Add a new list item and enter in a tweet and save.



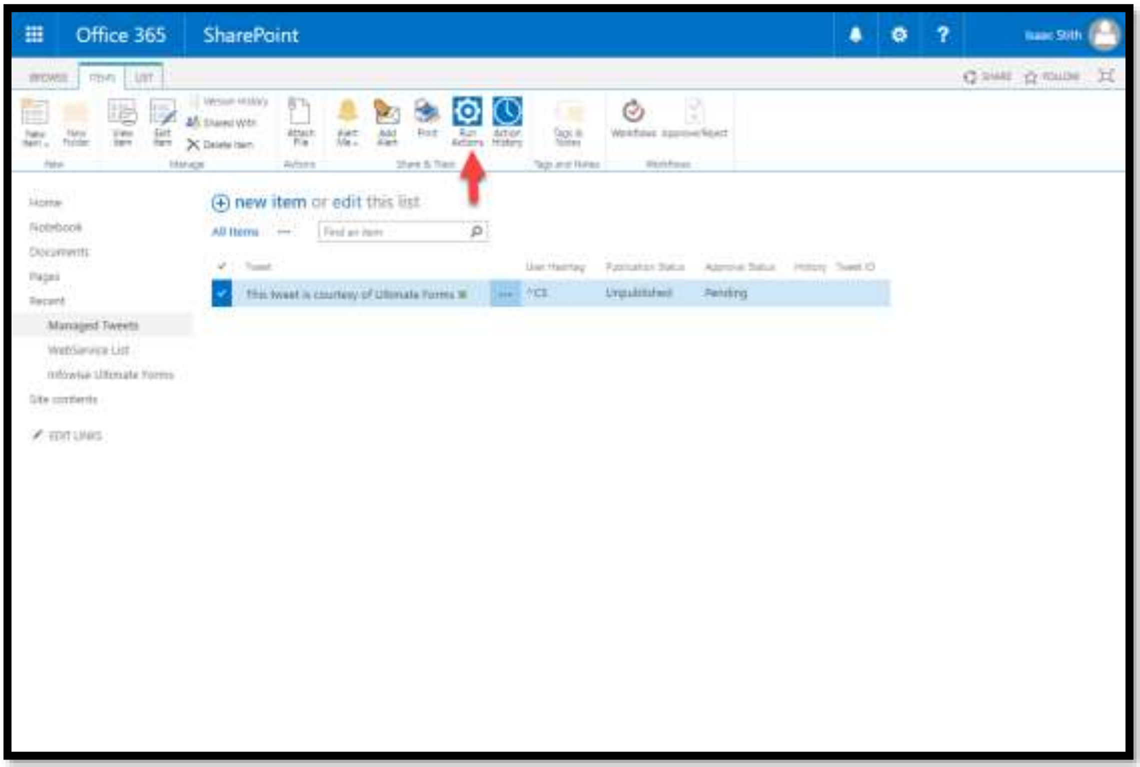
Managed Tweets

new item or edit this list

All Items ··· Find an item

✓	Tweet		User Hashtag	Publication Status	Approval Status	History	Tweet ID
	This tweet is courtesy of Ultimate Forms	···	^CS	Unpublished	Pending		

We see that the status is un approved and pending, now click the new item and on the Ribbon, go to Items, and Run Actions.

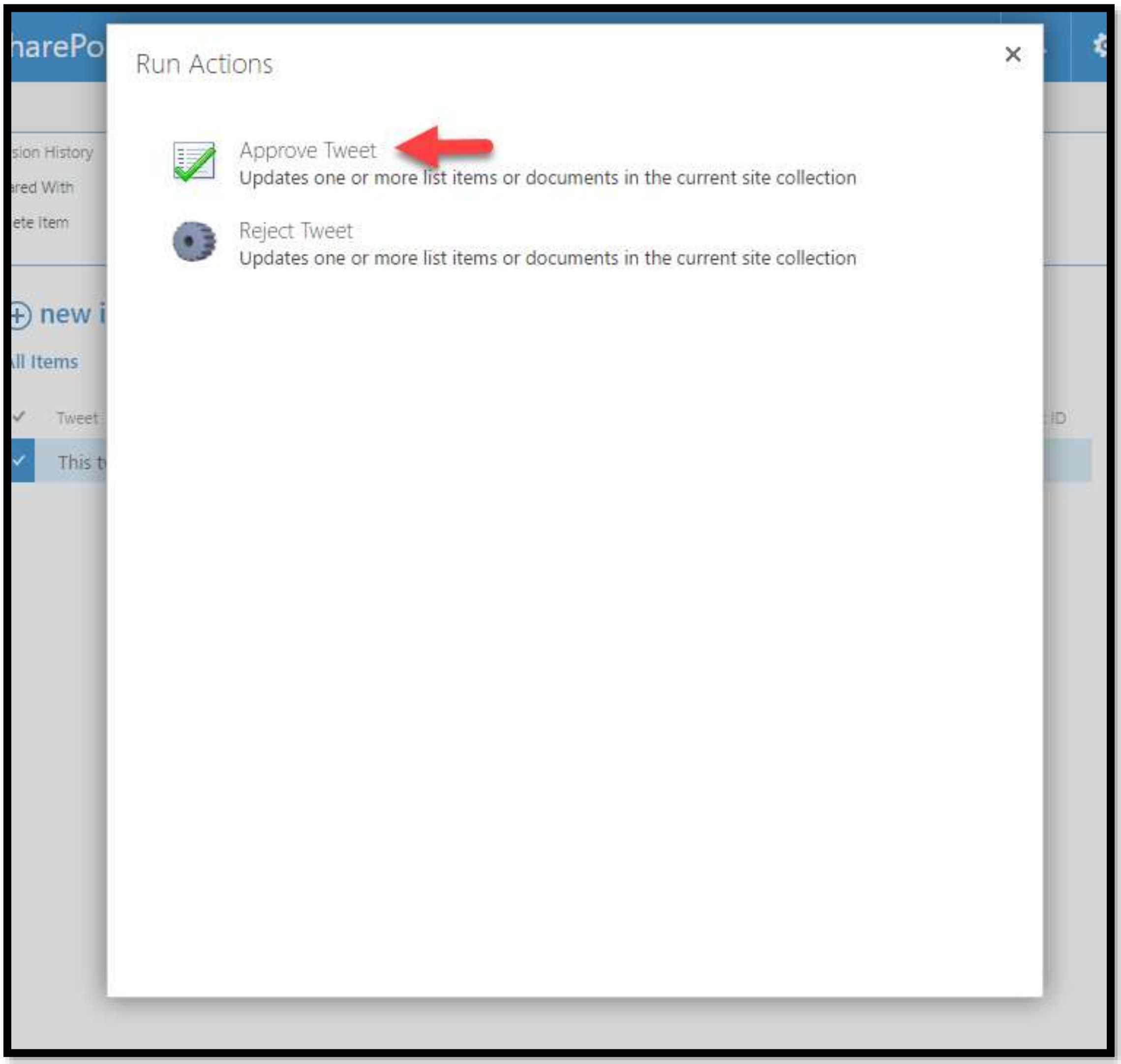


new item or edit this list

All Items ··· Find an item

✓	Tweet		User Hashtag	Publication Status	Approval Status	History	Tweet ID
✓	This tweet is courtesy of Ultimate Forms	···	^CS	Unpublished	Pending		

This should bring present the Approve or reject dialog

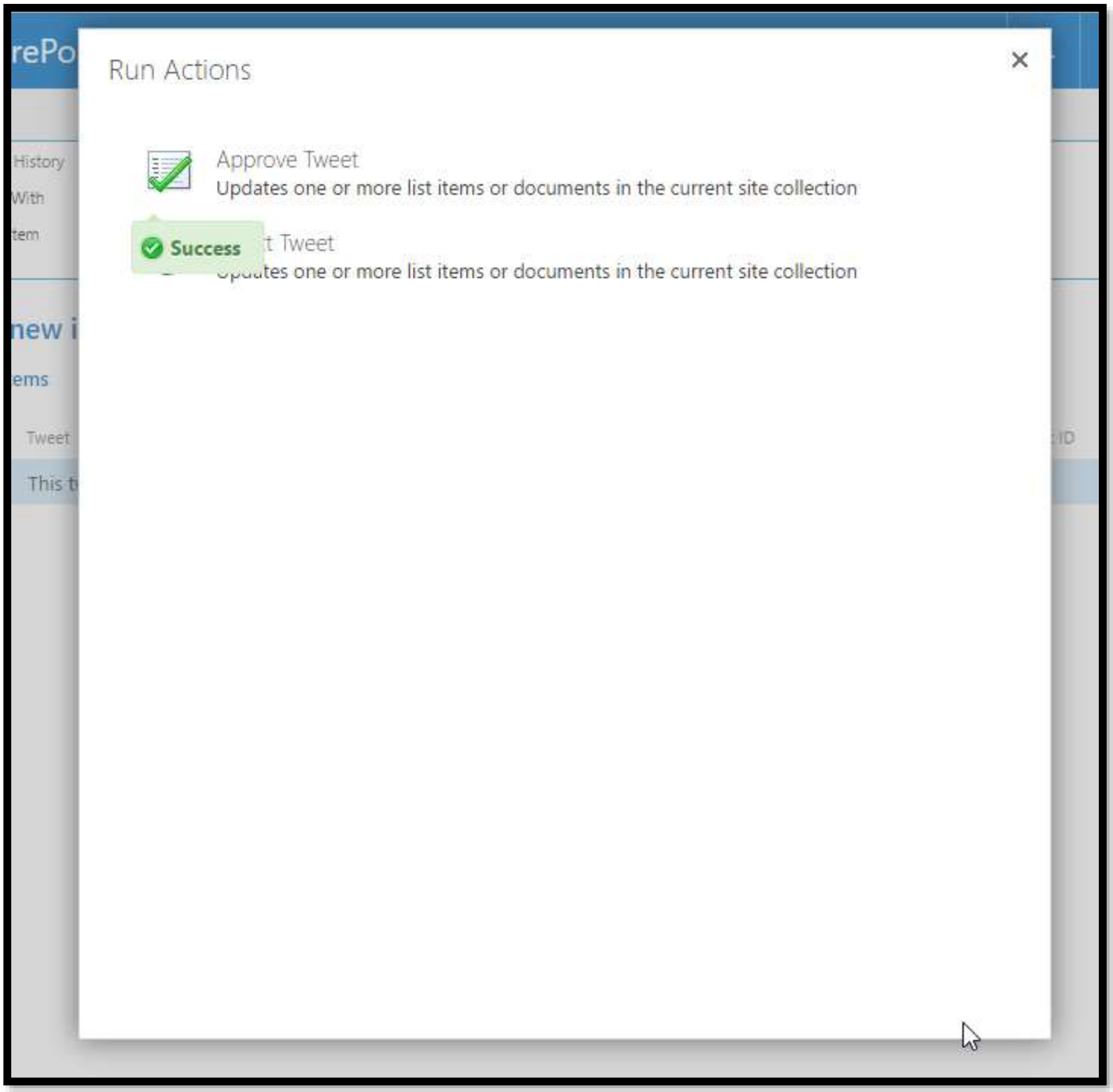


Run Actions

Approve Tweet
Updates one or more list items or documents in the current site collection

Reject Tweet
Updates one or more list items or documents in the current site collection

Click approve tweet



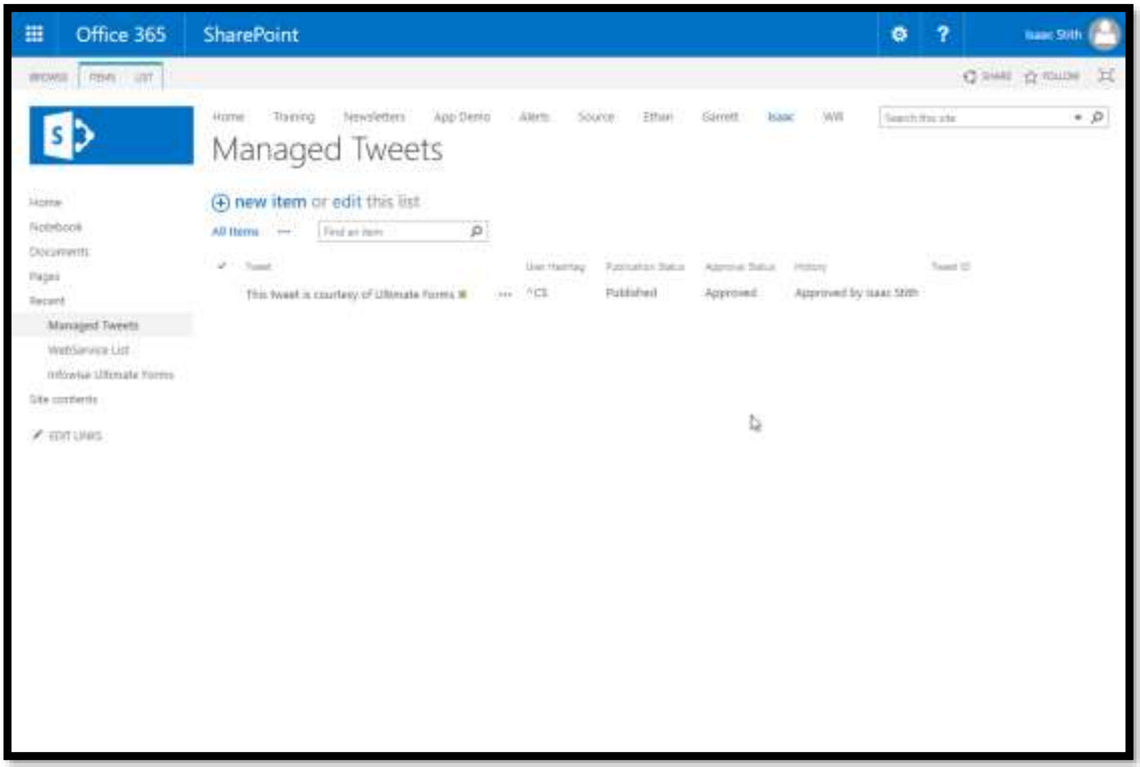
Run Actions

Approve Tweet
Updates one or more list items or documents in the current site collection

Success

t Tweet
ucates one or more list items or documents in the current site collection

Now the tweet should be published and when we go back to the list



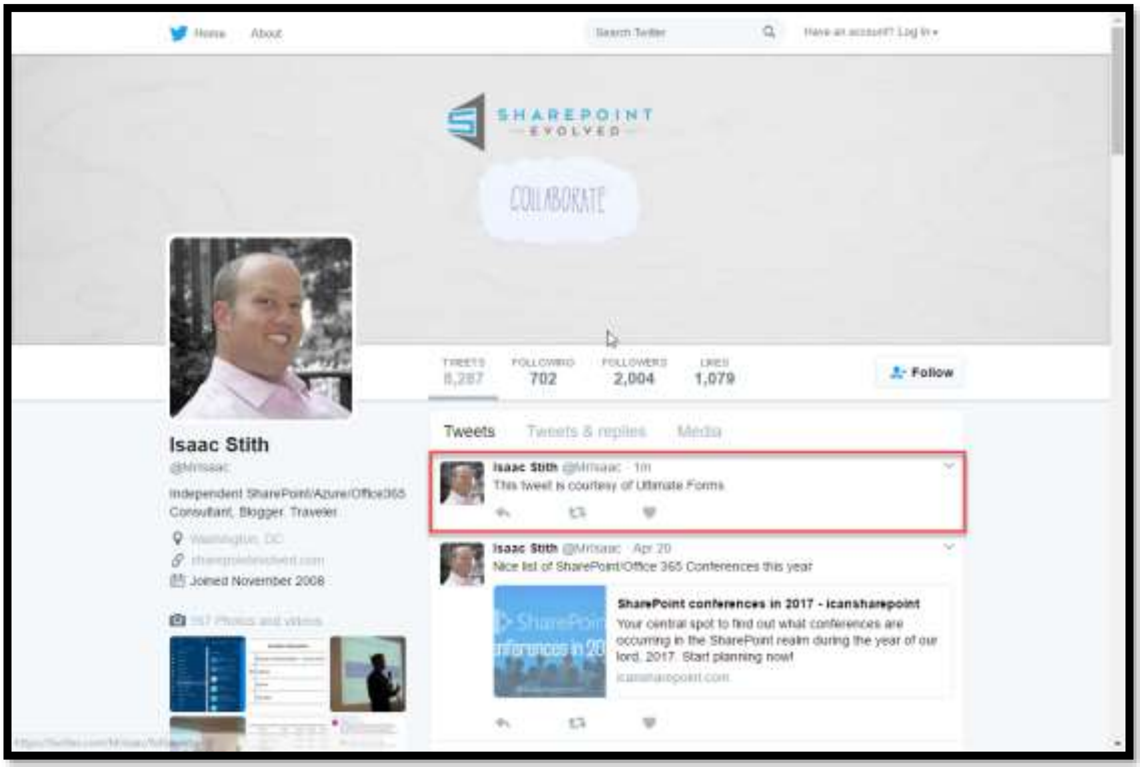
Managed Tweets

⊕ new item or edit this list

All Items ··· Find an item

✓	Tweet	User Hashtag	Publication Status	Approval Status	History	Tweet ID
	This tweet is courtesy of Ultimate Forms	^CS	Published	Approved	Approved by Isaac Stith	

It shows approved and published in the status columns. Now we need to check though if the tweet was sent by logging into twitter on the account you authorized in the Create Logic App step. Yyou can see below that the tweet sent successfully via Ultimate Forms.



Isaac Stith
@MrIsaac
Independent SharePoint/Azure/Office365 Consultant, Blogger, Traveler
Washington, DC
sharepointevolved.com
Joined November 2008
357 Photos and videos

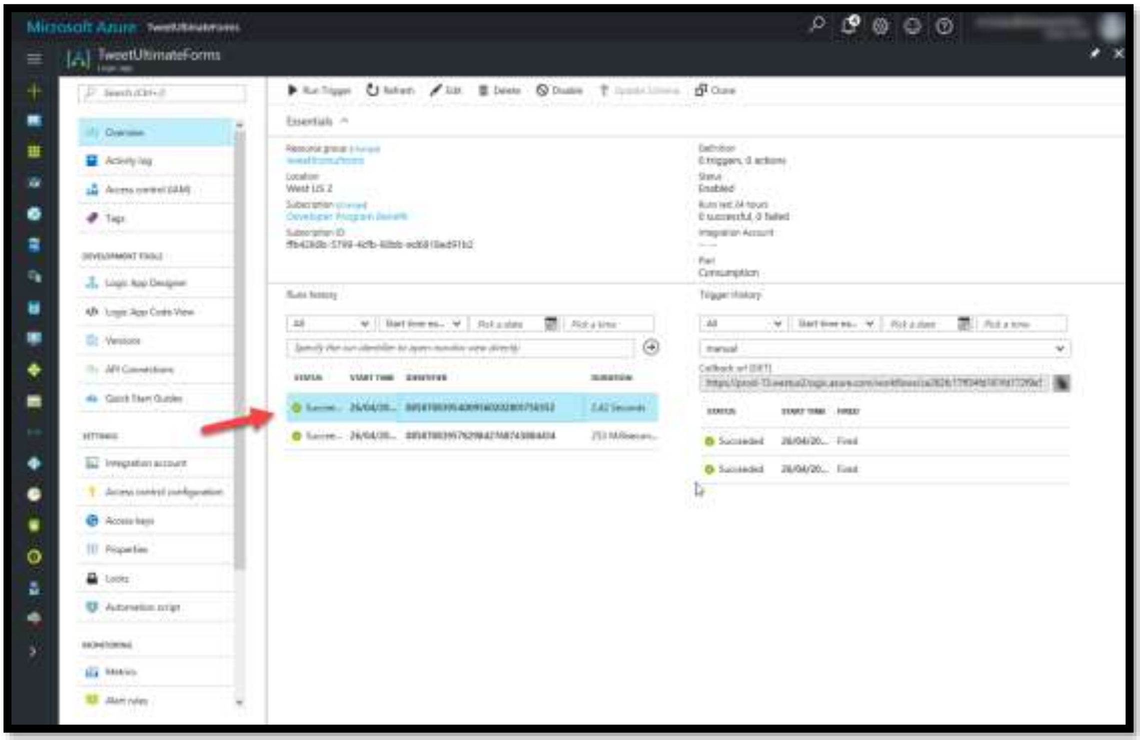
Isaac Stith @MrIsaac · 1m
This tweet is courtesy of Ultimate Forms

Isaac Stith @MrIsaac · Apr 20
Nice list of SharePoint/Office 365 Conferences this year

SharePoint conferences in 2017 - icansharepoint
Your central spot to find out what conferences are occurring in the SharePoint realm during the year of our lord, 2017. Start planning now!
icansharepoint.com

In Addition

What’s cool about logic apps, is that you can see each of the calls to the web service when the triggers run.



Resource group (change)
TweetUltimateForms

Location
West US 2

Subscription (change)
Developer Program Benefit

Subscription ID
ffb42b0b-5799-4cfb-60bb-ecb610ad91b2

Definition
0 triggers, 0 actions

Status
Enabled

Runs last 24 hours
0 successful, 0 failed

Integration Account

Plan
Consumption

Runs history

All | Start time ea... | Pick a date | Pick a time

Specify the run identifier to open monitor view directly

STATUS	START TIME	IDENTIFIER	DURATION
Succee...	26/04/20...	[illegible]	2.42 Seconds
Succee...	26/04/20...	[illegible]	753 Millisecon...

Trigger History

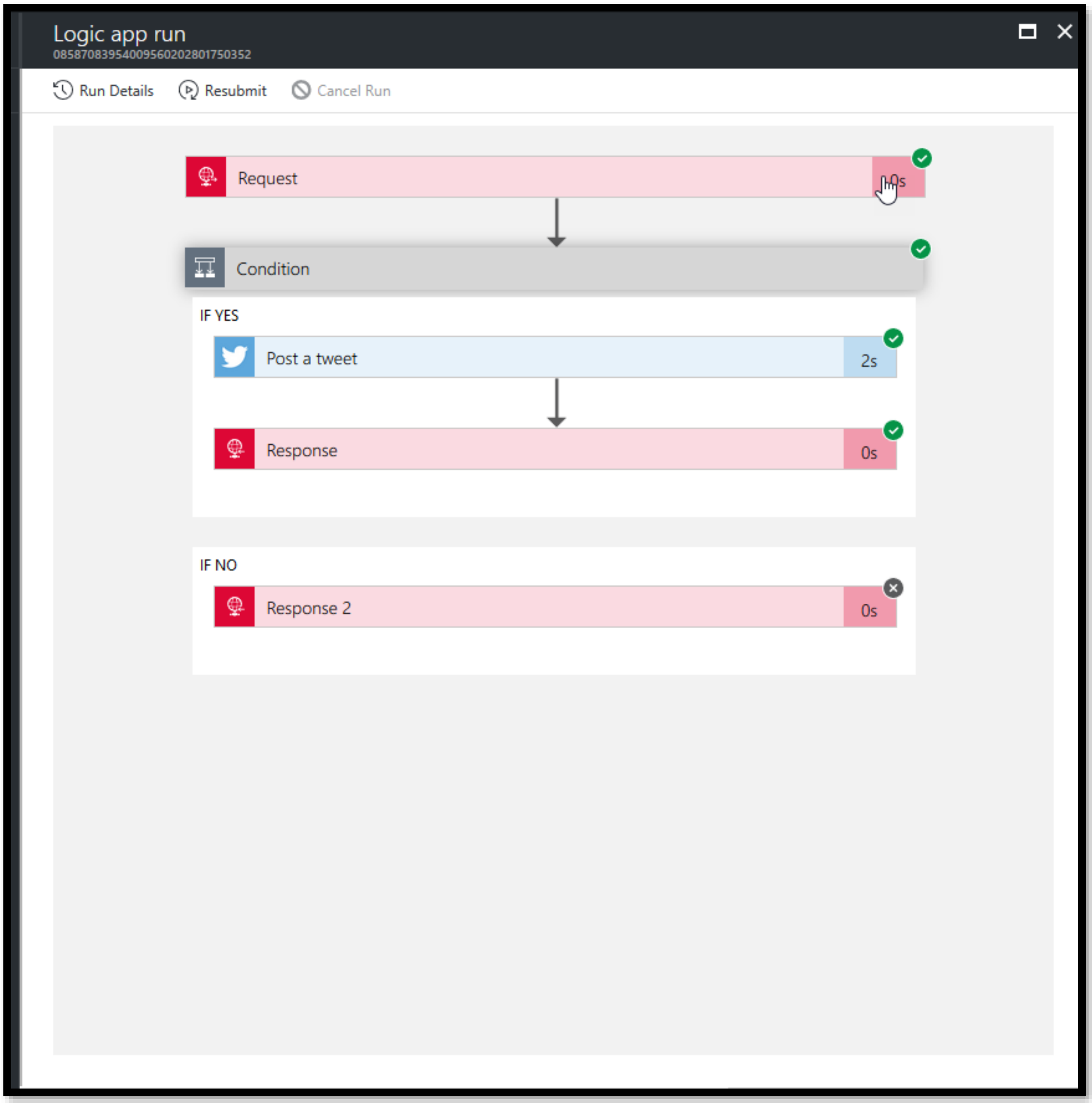
All | Start time ea... | Pick a date | Pick a time

manual

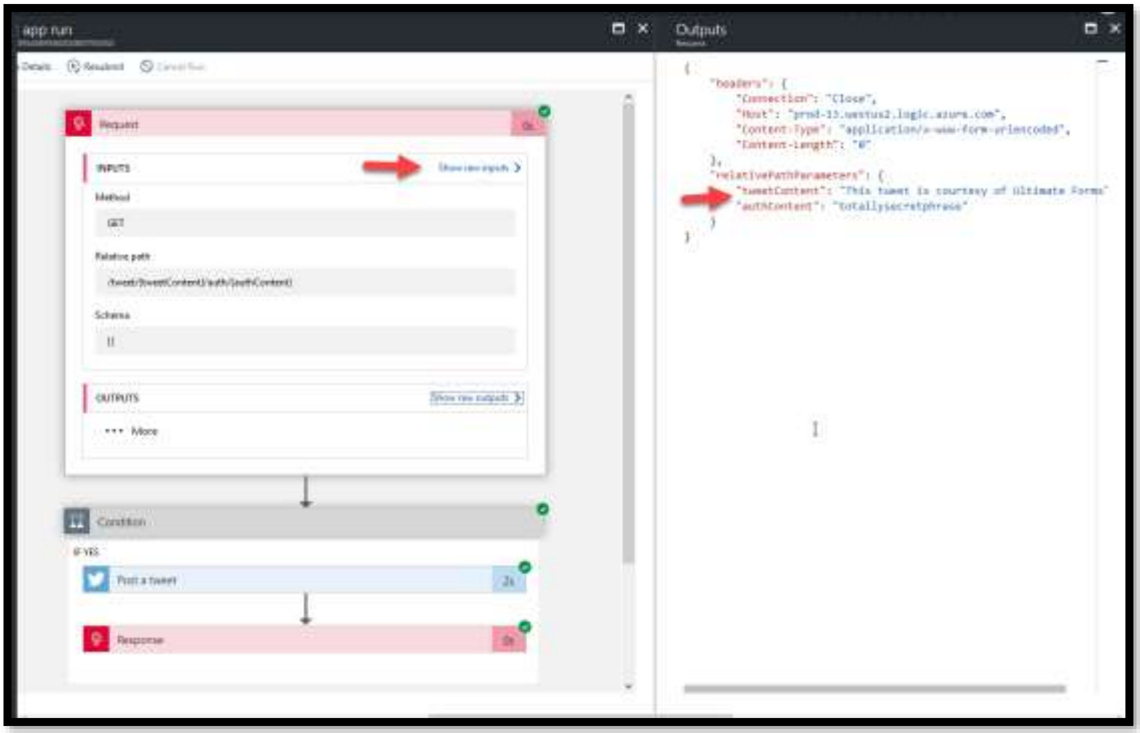
Callback url [GET]
https://prod-13.westus2.logic.azure.com/workflows/[illegible]

STATUS	START TIME	FIRED
Succeeded	26/04/20...	Fired
Succeeded	26/04/20...	Fired

So, if you click into this run you see the flow of the app as it executed for this particular request



And if you expand the sections you can see the debug or view the captured information for the request



Method
GET

Relative path
/tweet/{tweetContent}/auth/{authContent}

Schema
{}

```
{
    "headers": {
        "Connection": "Close",
        "Host": "prod-13.westus2.logic.azure.com",
        "Content-Type": "application/x-www-form-urlencoded",
        "Content-Length": "0"
    },
    "relativePathParameters": {
        "tweetContent": "This tweet is courtesy of Ultimate Forms"
        "authContent": "totallysecretphrase"
    }
}
```

For instance, you can see the specific tweet and auth phrase that was passed to this service under the parameters tweetContent and authContent.

For further reading on Azure Logic Apps, check out the documentation on Azure here.